Pràctica 5: Sessions

$_SESSION

Guillem Serrat Bahí

Index

VPS **4**
Introducció **5**
Pràctica 5.1 **6**
Objectiu de la pràctica 6
Demostració 7
Introducció a les sessions 8
Documentació del codi de l'aplicació 10
Pàgina inicial 10
Pàgina del cistell 11
Pràctica 5.2 **13**
Objectiu de la pràctica 13
Demostració 14
Estructura de directoris 15
Documentació del codi de l'aplicació 16
Creació de la base de dades i les seves taules 16
Connexions a la BD 18
Catàleg 19
Cistell 25
Gestió del cistell 30
Afegir productes des del catàleg 31
Afegir productes des del cistell 33
Retirar una quantitat de productes 35
Eliminar un producte del cistell 37
Buidar el cistell 38
Compra 39
Gestió del catàleg 42
Històric de compres 43
Afegir o retirar stock d'un producte 47
Afegir productes al catàleg 50
Funció per el preu dels productes i aplicar descomptes 52
Pràctica 5.3 **54**
Objectiu de la pràctica 54
Demostració 55
Estructura de directoris 56
Documentació del codi de l'aplicació 57
Creació de la BD i les seves taules 57
Connexions a la BD 59
Pàgina inicial 60
Registre d'usuaris 61
Inici de sessió 64

Inici de sessió amb 2FA 72
Recuperació de contrasenya 78
Pàgina privada 81
Edició de perfil d’usuari 84
Verificació de correu electrònic 92
Tauler de l’administrador 95
Logout 100
Funcions 101
encriptar_contrasenya 101
verificar_contrasenya 101
registrar_activitat 102
esta_autenticat 102
requerir_autenticacio 103
es_admin 103
es_admin_ed 104
Configuració del VPS per l'enviament de correus electrònics amb PHP 105
Configuració de Nominalia 105
Configuració del VPS 106
Configuració de PHP 108
Configuració del NSG d’Azure 109

VPS

Tant aquesta pràctica com la resta d'aquestes i altres projectes es poden consultar dins de la pàgina web www.gserrat.cat.

A més a més, a la pàgina web www.wiki.gserrat.cat es troba una wiki sobre la pràctica realitzada.

Enllaços ràpids:

- Pàgina inicial de la pràctica 5.1
- Documentació en format wiki de la pràctica 5.1
- Pàgina inicial de la pràctica 5.2
- Documentació en format wiki de la pràctica 5.2
- Pàgina inicial de la pràctica 5.3
- Documentació en format wiki de la pràctica 5.3

Introducció

Aquesta pràctica es centra en l'aplicació de sessions en PHP dins del desenvolupament d'aplicacions web, amb l'objectiu de posar en pràctica diferents funcionalitats que requereixen mantenir informació de l'usuari al llarg de la navegació.

Al llarg de la pràctica, les sessions s'apliquen en diferents contextos, com ara la gestió de dades entre pàgines, el funcionament d'un cistell de compra i la autenticació d'usuaris, combinant-les amb persistència en base de dades. Aquest enfocament permet integrar coneixements previs, com l'ús de consultes SQL i la gestió de dades d'usuari, ampliant les funcionalitats de les aplicacions desenvolupades.

Dins de la documentació, s'ometen moltes parts de codi que realment ja han estat estudiades amb anterioritat en anteriors pràctiques i projectes, o funcionalitats de la programació bàsica (com condicionals, comparacions, tipus de funcions, etc)

Dins la pràctica 5.1 és documenta el funcionament i la implementació de les sessions. A la pràctica 5.2 i 5.3 no es torna a documentar, incorporant únicament a la documentació la funcionalitat de les sessions per aquell codi

Per últim, cal remarcar que l'activitat està orientada a PHP i per tant no és mostrarà cap element HTML ni CSS de cap codi a menys que sigui necessari per entendre certs aspectes de PHP.

Pràctica 5.1

Objectiu de la pràctica

L'objectiu d'aquesta pràctica és fer una introducció a les sessions i aprendre els seus fonaments de funcionament i ús.

Per posar en producció aquests coneixements, es realitzarà una pàgina web on l'usuari haurà d'introduir el seu nom i cognom i afegir uns elements a un cistell.

Un cop els introdueixi, en cas que l'usuari vulgui anar al cistell, mitjançant sessions haurem d'emmagatzemar la informació de la pàgina inicial (nom i cognom i elements al cistell) per mostrar-los al cistell.

Dins del cistell es podrà modificar els elements afegits al cistell però no el nom i cognom. A més, s'habilitarà la funcionalitat de tancar la sessió per buidar tota la informació i poder introduir de nou el nom i cognom i altres elements

Demostració

En el següent vídeo es pot comprovar una prova de funcionament de l'aplicació:

demostracioA5.1.mp4

Introducció a les sessions

Sempre que en una pàgina web s'incloguin sessions, s'ha d'iniciar la sessió.

```
session_start();
```

Les sessions són un mecanisme per desar dades de l'usuari al servidor. Per treballar amb sessions es fa servir la variable global $_SESSION

La variable $_SESSION és una array, per tant es pot definir vectors (camps) amb valors (dades). Aquests camps poden ser valors únics o més arrays.

Un exemple de model de dades d'una sessió és el següent:

```
$_SESSION['usuari'] = [
    'nom'       => 'Joan',
    'cognom'    => 'García',
    'edat'      => 28,
    'ingredients' => [
        'Tomàquet',
        'Llevat',
        'Galeta'
    ]
];

$_SESSION['data_hora'] = [
    'data'       => '12/1/2026',
    'hora'    => '15:04',
];
```

Podem comprovar que la sessió té dos grans grups de dades: usuari i data_hora. Aquests "grups de dades" són vectors de la sessió, que a la vegada també son arrays amb més vectors

> Per crear un objecte de la sessió ho fem de la mateixa manera que afegint un vector a una array normal, amb la diferència que especifiquem $_SESSION com array

```
$_SESSION[<nomObjecte>] = <valor>;
$_SESSION['nom'] = $_POST['nom']; // "Nom", tindrà el valor que s'ha enviat al formulari

$_SESSION['nom'] = "Guillem";
echo $_SESSION['nom']; // Imprimirà "Guillem"
```

L'objectiu de les sessions és mantenir les mateixes dades entre les pàgines navegades. En el moment que es desa una informació a la sessió, totes les pàgines que inicïin la sessió podran recuperar aquella informació i fer-la servir

Pàgina 1

```
session_start();
$_SESSION['nom'] = "Guillem";
echo $_SESSION['nom']; // Imprimirà "Guillem"
```

Pàgina 2

```
session_start();
echo $_SESSION['nom']; // Imprimirà "Guillem"
```

Pàgina 3

```
echo $_SESSION['nom']; // Donarà error, ja que no hem iniciat la sessió
```

A l'hora de modificar les dades d'una sessió és exactament la mateixa gestió que amb una array normal

En el moment que volem eliminar un objecte de la sessió, farem servir la funció unset()

```
unset($_SESSION["nom"]);
```

Per últim, per tancar per complet la sessió i eliminar totes les dades dins d'aquesta, realitzarem el següent procediment:

Primer esborrarem les dades de la sessió

```
$_SESSION = array();
```

Seguidament, destruirem la pròpia sessió

```
session_destroy();
```

Per últim, esborrarem la cookie PHPSESSID del navegador

```
setcookie(session_name(), '', time()-3600, '/');
```

Documentació del codi de l'aplicació

Pàgina inicial

A la pàgina inicial es mostrarà 2 inputs per introduir el nom i cognom, a més de 4 elements per afegir al cistell amb caselles de selecció

Primerament, iniciarem la sessió

```php
<?php
// Iniciem la sessió
session_start();
```

En cas de respondre el formulari, desarem les variables nom i cognom a la sessió

```php
if ($_SERVER['REQUEST_METHOD'] === 'POST') { // Quan es respongui el formulari
    $_SESSION['nom_usuari'] = trim(string: $_POST['nom']); // Desem el nom de l'usuari a la sessió
    $_SESSION['cognom_usuari'] = trim(string: $_POST['cognom']); // Desem el cognom de l'usuari a la sessió
```

Seguidament, crearem l'array de regals a la sessió i emmagatzemarem els regals seleccionats en aquesta array

```php
// Desa regals seleccionats
$_SESSION['regals'] = array(); // Creem l'objecte de la sessió "regals", que és una array
if (isset($_POST['regals']) && is_array(value: $_POST['regals'])) { // Comprovem que l'array regals existeix
    $_SESSION['regals'] = $_POST['regals']; // Introduim a l'array regals els regals del formulari (els regals
}
```

Finalment, redirigeix a l'usuari a la pàgina del cistell

```php
// Redirecció al cistell
header(header: 'Location: cistell.php');
exit;
```

Pàgina del cistell

Dins la pàgina del cistell, iniciarem sessió per recuperar les dades de la sessió.

```php
<?php
// Iniciem la sessió
session_start();
```

El següent pas serà comprovar que el nom i el cognom de l'usuari estan definits (és a dir, a la sessió, ha entrat a la pàgina inicial i ha introduit el seu nom i cognom). En cas que no ho estiguin, redirigirem l'usuari a la pàgina inicial

```php
// Comprovem que el nom i cognom de l'usuari estigui definit
if (!isset($_SESSION['nom_usuari']) || !isset($_SESSION['cognom_usuari'])) {
    header(header: 'Location: index.php'); // En cas que no s'hagi definit el nom i cognom, redirigirem a l'usuari a l'index
    exit;
}
```

Recuperarem les dades de la sessió i les desarem a variables

```php
$nom_complet = htmlspecialchars(string: $_SESSION['nom'] . ' ' . $_SESSION['cognom']);
$regals = isset($_SESSION['regals']) ? $_SESSION['regals'] : array(); // Recuperem els
```

Dins de l'html, mostrarem el nom i cognom

```php
<body>
    <h1>Cistell de <?php echo $nom_complet; ?></h1>
```

Mitjançant un condicional, comprobarem si el cistell té algun element

```php
<h2>Regals seleccionats:</h2>
<?php if (empty($regals)): // En cas que no hi hagi regals?>
    <p>No heu seleccionat cap regal</p> <!-- Mostrarem que no hi ha cap regal -->
<?php else: ?>
```

En cas que hi hagi un element, per cada element dins del cistell es mostrarà com a item de la llista

```php
<?php else: ?>
    <ul>
        <?php foreach ($regals as $regal): // Per cada regal al cistell?>
            <li><?php echo htmlspecialchars(string: $regal); // El mostrem com a objecte d'una llista?></li>
        <?php endforeach; ?>
    </ul>
<?php endif; ?>
```

Més endavant tenim la possibilitat d'actualitzar el cistell. Primer definirem les possibles opcions per actualitzar el cistell, i per cada opció, imprimirem una casella de selecció. En cas que l'element estigui dins del cistell, aquesta casella estarà ja marcada

```php
<?php
$opcions = array('telefon', 'cotxe', 'teatre', 'cine'); // Definim les opcions per actualitzar regals
foreach ($opcions as $opcio): // Per cada opció
    $checked = in_array(needle: $opcio, haystack: $regals) ? 'checked' : ''; // En cas que el regal ja e
?>
    <label>
        <input type="checkbox" name="regals[]" value="<?php echo $opcio; ?>" <?php echo $checked; ?>> <!
        <?php echo $opcio; ?>
    </label><br>
<?php endforeach; ?>
```

Tots les inputs comparteixen "nom", una array. Per tant, si una casella de verificació està marcada al enviar el formulari, el seu valor (el nom del producte) s'afegirà a aquesta array

Quan premem el botó d'actualitzar el cistell, tornarem a fer el mateix que a la pàgina inicial, esborrarem tota l'array de regals i afegirem a l'array de la sessió els valors de l'array del formulari

```php
// Actualitza regals si s'ha tramès el formulari
if ($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['actualitzar'])) {
    $_SESSION['regals'] = array(); // Creem l'objecte de la sessió "regals", que és una array
    if (isset($_POST['regals']) && is_array(value: $_POST['regals'])) { // Comprovem que l'array regals existeix
        $_SESSION['regals'] = $_POST['regals']; // Introduim a l'array regals els regals del formulari (els regals
    }
}
```

Per últim, si decidim tancar la sessió mitjançant el botó de tancar sessió, realitzarem el procés de tancament de sessió.

```php
// Finalització de sessió
if (isset($_GET['finalitzar'])) { // Si es tanca la sessió
    $_SESSION = array();
    if (isset($_COOKIE[session_name()])) { // Verifiquem que la galeta de sessió s'elimini
        setcookie(name: session_name(), value: '', expires_or_options: time()-3600, path: '/');
    }
    session_destroy();
    header(header: 'Location: index.php');
    exit;
}
```

Pràctica 5.2

Objectiu de la pràctica

L'objectiu de la pràctica és realitzar una pàgina web de compra de productes amb diferents funcionalitats, tenint com a base un catàleg i un cistell de productes.

S'implementa un catàleg de productes, on es mostren tots els productes amb els seus detalls: nom, categoria, preu, stock, etc. A cada producte es pot escollir quina quantitat afegir al cistell. Quan s'afegeix un producte al cistell es desa en una sessió

A més a més, a l'icona del cistell es pot comprovar les unitats de productes que hi ha dins

També s'implementa un cistell, el qual mostra els productes i el seu subtotal, les unitats i el preu unitari a partir de les dades de la sessió activa. Es dóna la possibilitat d'afegir més quantitat, retirar-ne o eliminar el producte per complet des del propi cistell per no haver de tornar al catàleg.

Al propi cistell també es dóna la possibilitat de buidar-lo per complet per tancar la sessió

Com a funcionalitats extra s'ha implementat:

- Implementació de categories i imatges als productes
- Possibilitat de cerca de productes segons categoria o nom dins del catàleg
- Control i verificació exhaustiu d'stock abans de realitzar operacions
- Aplicació de descomptes segons quantitat d'unitats
- Possibilitat de compra de productes i registre en un històric
- Administració del catàleg
    - Espai protegit amb usuari i contrasenya per permetre l'accés a usuaris autoritzats
    - Consulta de l'històric de compres
    - Modificació de l'stock dels productes
    - Inserció de nous productes al catàleg

Demostració

En el següent vídeo es pot comprovar una prova de funcionament de l'aplicació:
demostracioA5.2.mp4

Estructura de directoris

L'estructura de directoris per aquesta pràctica és la següent.

5.2

```
├── administracioBD (Protegit amb usuari i contrasenya)
│   ├── creacioBDiTaula.php
│   ├── eliminacioBD.php
│   └── veureErrors.txt
├── administracioCataleg (Protegit amb usuari i contrasenya)
│   ├── administracioCataleg.php
│   ├── afegirProductes.php
│   ├── afegirStock.php
│   └── historialCompres.php
├── cataleg.php
├── cistell.php
├── compra.php
├── connexioBD
│   ├── connexioR.php
│   └── connexioRW.php
├── css
│   ├── stylesAfegirProductes.css
│   ├── stylesCataleg.css
│   ├── stylesCistell.css
│   ├── stylesCompra.css
│   ├── stylesHistorial.css
│   └── stylesStock.css
├── documentacio.txt
├── funcions.php
├── imatges
│   ├── altaveu.jpg
│   ├── auriculars.jpg
│   ├── monitor.jpg
│   ├── portatil.jpg
│   ├── ratoli.jpg
│   ├── teclat.jpg
│   └── webcam.jpg
└── processa.php
```

Documentació del codi de l'aplicació

Creació de la base de dades i les seves taules

Aquesta aplicació tindrà una BD anomenada "botiga" amb codificació UTF8 amb tres taules:

- Taula per desar els productes i els seus detalls (nom "productes")
    - Nom
    - Preu
    - Descripció
    - Estoc
    - Categoria
    - Imatge del producte

```php
// Creem la taula productes, on s'emmagatzemen els productes per vendre al catàleg
$sql = "CREATE TABLE productes (
id INT AUTO_INCREMENT PRIMARY KEY,
nom VARCHAR(100) NOT NULL,
preu DECIMAL(10, 2) NOT NULL,
descripcio TEXT,
estoc INT NOT NULL DEFAULT 1,
categoria VARCHAR(30) NOT NULL,
imatge VARCHAR(30) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;";

$conn->exec(statement: $sql);
```

- Taula per desar les compres (nom "compres")
    - Total de compra
    - Data de compra

```php
// Creem la taula on es registrarà les compres (únicament la data de la compra i el total d'aquesta)
// Posteriorment, es crea una taula per desar-hi els detalls
$sql = "CREATE TABLE compres (
    id INT AUTO_INCREMENT PRIMARY KEY,
    data_compra DATETIME NOT NULL,
    total float NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;";
```

- Taula per desar els detalls de les compres realitzades (nom “compres_detall”)
    - Productes comprats
    - Quantitat
    - Preu de cada producte i total de productes
    - ID de la compra a la que corresponen dels detalls (de la taula compres)
    - ID del producte al que correspon preu, quantitat, etc (de la taula productes)

```
// Creem la taula per desar els detalls de les compres. Desarem de cada compra quin productes ha comprat i els seus detalls
// Relacionarem ID de compra amb la taula compres per identificar la compra a la que correspon el detall del producte
// Relacionarem la ID de producte amb la taula productes per identificar quin producte és
$sql = "CREATE TABLE compres_detall (
    id INT AUTO_INCREMENT PRIMARY KEY,
    compra_id INT NOT NULL,
    producte_id INT NOT NULL,
    quantitat INT NOT NULL,
    preu_unitari DECIMAL(10,2) NOT NULL,
    total_producte float NOT NULL,

    FOREIGN KEY (compra_id) REFERENCES compres(id),
    FOREIGN KEY (producte_id) REFERENCES productes(id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;";
```

A més, s’assignen els següents permisos a la BD:

- Usuari iot
    - Escriptura i lectura
- Usuari convidat
    - Lectura

```
// Afegim permissos a diversos usuarios sobre la BD
$permisosRW = "GRANT SELECT, INSERT, UPDATE, DELETE ON botiga.* TO 'iot'@'localhost';";
$permisosR = "GRANT SELECT ON botiga.* TO 'convidat'@'localhost';";

$conn->exec(statement: $permisosRW);
$conn->exec(statement: $permisosR);
$conn->exec(statement: "FLUSH PRIVILEGES;");
```

Connexions a la BD

Les dues connexions a la BD es diferencien segons els permisos aplicats anteriorment: de lectura i escriptura i únicament de lectura.

La connexió de lectura la qual únicament s'usa en la pàgina del catàleg per mostrar productes, és la següent:

```php
<?php
    $servidor = "127.0.0.1";
    $usuari = "convidat";
    $contrasenya = "benvingut";
    $nomDB = "botiga";
    $opcions = array(
    PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION,
    PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,
    PDO::MYSQL_ATTR_INIT_COMMAND => "SET NAMES utf8mb4"
    );

    try {
        $pdo = new PDO(dsn: "mysql:host=$servidor;dbname=$nomDB", username: $usuari, password: $contrasenya, options: $opcions);
    } catch(PDOException $e) {
        echo "No es pot connectar. Motiu: " . $e->getMessage();
    }
?>
```

La connexió de lectura i escriptura, que s'usa pràcticament en la resta de pàgines, és la següent:

```php
<?php
    $servidor = "127.0.0.1";
    $usuari = "iot";
    $contrasenya = "iot";
    $nomDB = "botiga";
    $opcions = array(
    PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION,
    PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,
    PDO::MYSQL_ATTR_INIT_COMMAND => "SET NAMES utf8mb4"
    );

    try {
        $pdo = new PDO(dsn: "mysql:host=$servidor;dbname=$nomDB", username: $usuari, password: $contrasenya, options: $opcions);
    } catch(PDOException $e) {
        echo "No es pot connectar. Motiu: " . $e->getMessage();
    }
?>
```

Catàleg

La primera pàgina en mostrar-se és el catàleg de productes.

Primer de tot, s'inicialitza la sessió i es requereix la connexió d'escriptura

```php
<?php
// Iniciem la sessió i requerim la connexió a la BD
session_start();
require_once "./connexioBD/connexioR.php";
```

Seguidament, en cas que no ho estigui, inicialitzem el cistell dins de la sessió, el qual és una array

```php
// Inicialitzar cistell
if (!isset($_SESSION['cistell']) || !is_array(value: $_SESSION['cistell'])) { // Si el cistell no està inicialitzat o no és una array
    $_SESSION['cistell'] = array();
}
```

Seguidament definim el comptador del cistell. Per defecte és 0, però si hi ha productes al cistell, per cada producte del cistell afegim la seva quantitat al comptador del cistell.

Verifiquem que el producte és una array (ja que conté els detalls com el nom, el preu, i el més important, la quantitat) i verifiquem que el vector "quantitat" estigui definit, a més d'assegurar que sigui un enter

```php
// Comptador de productes al cistell
$comptador = 0; // El comptador inicial és 0
foreach ($_SESSION['cistell'] as $prod) { // Per cada producte
    if (is_array(value: $prod) && isset($prod['quantitat'])) {
        $comptador = $comptador + (int)$prod['quantitat']; // A
    }
}
```

Seguidament farem la consulta dels productes, la qual pot tenir tres possibilitats:

- S'ha realitzat una cerca per nom
- S'ha realitzat una cerca per categoria
- No s'ha realitzat cap cerca (primera vegada en la pàgina)

En cas que s'hagi realitzat una cerca, ho sabrem ja que s'ha enviat un formulari GET amb el camp "nom" o el camp "categoria"

En cas que es faci una cerca per nom, es farà el següent:

```php
// Filtrar per nom
if (!empty($_GET['nom'])) { // En cas que s'hagi rebut algun valor pel camp "nom" (i per tant fer una cerca per nom)
    $cerca = "nom"; // Definim que a la consulta SQL es filtrarà per nom
    $valor = "%" . $_GET['nom'] . "%"; // Afegim els % degut a que és sintaxis de les consultes SQL
    $sql = "SELECT * FROM productes WHERE $cerca LIKE :valor"; // Exemple: SELECT * FROM productes WHERE nom LIKE %pc%
    $stmt = $pdo->prepare(query: $sql);
    $stmt->bindParam(param: ':valor', var: &$valor);
    $stmt->execute();
```

En cas que sigui una cerca per categoria, es farà el següent:

```php
// Filtrar por categoría
} elseif (!empty($_GET['categoria'])) { // En cas que s'hagi rebut algun valor pel camp "categoria" (i per tant fer una cerca per cate
    $cerca = "categoria"; // Definim que a la consulta SQL es filtrarà per categoria
    $valor = "%" . $_GET['categoria'] . "%"; // Afegim els % degut a que és sintaxis de les consultes SQL
    $sql = "SELECT * FROM productes WHERE $cerca LIKE :valor"; // Exemple: SELECT * FROM productes WHERE categoria LIKE %periferic%
    $stmt = $pdo->prepare(query: $sql);
    $stmt->bindParam(param: ':valor', var: &$valor);
    $stmt->execute();
```

I, per defecte, en cas que no es faci cap cerca, es recuperaran tots els productes de la BD

```php
// En cas que no s'hagi enviat cap camp del formulari (no hi ha cap cerca)
} else {
    $sql = "SELECT * FROM productes"; // Consulta per defecte, tots els productes
    $stmt = $pdo->prepare(query: $sql);
    $stmt->execute();
}
```

Per últim, desarem el fetch de la consulta SQL a la variable $productes

```php
$productes = $stmt->fetchAll();
```

$productes conté els productes amb els seus detalls

A partir d'aquí, comença l'HTML, on es mostrarà el contingut de la pàgina. Primerament es mostren dos botons: un per administrar el catàleg i l'altre per anar al cistell

```
<!-- Botó per anar al cistell -->
<p>
    <a href="cistell.php" class="cart-button">
    <i class="fas fa-shopping-cart"></i>
    <span class="cart-count"><?php echo $comptador; ?></span>
    </a>
</p>

<!-- Botó administració del catàleg -->
<a href="./administracioCataleg/administracioCataleg.php" class="admin-link">
    Administració del catàleg
</a>
```

Seguidament, es mostra el formulari de cerca

```
<!-- Formulari de cerca -->
<form method="get" action="cataleg.php">
    <fieldset>
        <legend>Cerca de productes</legend>

        <label>
            Nom:
            <input type="text" name="nom"
                value="<?php // En cas de tenir un valor del formulari (refresquem la pàgina amb el formulari enviat)
                if (isset($_GET['nom'])) { // El camp tindrà ja definida la cerca especificada anteriorment
                    echo htmlspecialchars(string: $_GET['nom']);
                } else { // En cas contrari, estarà buit
                    echo '';
                }?>">
        </label>

        <label>
            Categoria:
            <input type="text" name="categoria"
                value="<?php // En cas de tenir un valor del formulari (refresquem la pàgina amb el formulari enviat)
                if (isset($_GET['categoria'])) { # El camp tindrà ja definida la cerca especificada anteriorment
                    echo htmlspecialchars(string: $_GET['categoria']);
                } else { // En cas contrari, estarà buit
                    echo '';
                }?>">
        </label>

        <input type="submit" value="Cerca">

        <a href="cataleg.php">
            <button type="button">Esborrar cerca</button>
        </a>
    </fieldset>
</form>
```

Per últim, es defineix un espai per mostrar missatges de confirmació o error. Aquests missatges únicament surten si estan definits, i es defineixen al codi de Gestió del cistell segons el resultat de l'operació realitzada

```php
<!-- Missatge de confirmació d'accions -->
<center>
<?php if (isset($_SESSION['missatge_ok'])): // Quan s'enviï un missatge de confirmació?>
    <p style="color: green; font-weight: bold;">
        <?php
            echo htmlspecialchars(string: $_SESSION['missatge_ok']); // Mostrem el missatge
            unset($_SESSION['missatge_ok']); // Eliminem el missatge, així al recarregar la pàgina no tornarà a sortir,
        ?>
    </p>
<?php endif; ?>
</center>

<!-- Missatge d'error -->
<center>
<?php if (isset($_SESSION['missatge_error'])): // Quan s'enviï un missatge d'error?>
    <p style="color: red; font-weight: bold;">
        <?php
            echo htmlspecialchars(string: $_SESSION['missatge_error']); // Mostrem el missatge
            unset($_SESSION['missatge_error']); // Eliminem el missatge, així al recarregar la pàgina no tornarà a sorti
        ?>
    </p>
<?php endif; ?>
</center>
```

A partir d'aquest moment, comença la mostra de productes, el qual varia segons la consulta SQL realitzada a partir de la cerca de productes.

Per això, després de comprovar que hi ha productes a mostrar, realitzem un foreach a l'array de productes

```php
<!-- Mostra de productes -->
<!-- En cas que la consulta realitzada anteriorment no tingui resultats (generalment per una cerca no correcte)-->
<?php if (empty($productes)): ?>
    <p>No s'han trobat productes.</p>
<?php endif; ?>

<!-- En cas que la consulta realitzada anteriorment hagi obtingut resultats -->
<div class="catalog">
<?php foreach ($productes as $p): ?> <!-- Per cada producte obtingut a la consulta -->
```

A partir d'aquí, per cada producte es farà exactament el mateix, fins fer-ho amb tots els productes.

Primer gestionarem la visualització de l'estoc. És important entendre el següent:

Suposem que a la base de dades tenim 10 unitats en estoc i afegim 2 productes al cistell:

- A la base de dades encara apareixen 10 unitats, perquè la compra no s'ha completat.
- Tot i així, només queden 8 unitats disponibles per afegir al cistell, ja que 2 ja estan "reservades" dins del cistell.

Si només ens guiéssim per l'stock de la base de dades, podríem continuar afegint productes al cistell de manera indefinida fins a l'hora de la compra, cosa que generaria un problema si l'estoc real ja no és suficient.

És per això que hem de definir un "stock visible", per evitar que els usuaris afegeixin mes productes dels possibles al cistell, però sense tocar la BD, en cas que es buidi el cistell.

Per això primer hem de saber si dins del cistell es troba el mateix producte (mitjançant una cerca per ID de producte). En cas afirmatiu, desem a una variable la quantitat que hi ha dins del cistell

```php
<?php
    // Estoc reservat al cistell
    $reservat = 0;
    if (isset($_SESSION['cistell'][$p['id']])) { // Dintre del cistell, comprova si algun dels productes té la mateixa ID que els productes retornats a la consulta SQ
        $reservat = (int)$_SESSION['cistell'][$p['id']]['quantitat']; // El reservat és, dins del cistell, la quantitat d'aquell producte
    }
```

Seguidament definim l'stock visible, el qual és l'stock que hi ha a la BD menys l'stock reservat dins del cistell

```php
$estoc_visible = $p['estoc'] - $reservat; // L'estoc visible (que no real) del producte és l'stock real menys al reservat
// Així, si hi ha productes al cistell, els restem de l'stock, però no a la BD, ja que es possible que es retirin productes
```

L'stock visible és la quantitat de productes que l'usuari pot afegir al cistell.

Si hi ha 10 a la BD i 0 al cistell: Stock visible = 10
Si hi ha 10 a la BD i 5 al cistell: Stock visible = 5
Si hi ha 10 a la BD i 8 al cistell stock visible = 2

L'stock de la BD es resta segons els productes del cistell. En el moment de fer la compra final, aquesta resta es fa a la BD en sí i s'actualitza l'stock

Un cop calculat l'stock visible, mostrarem els detalls del producte

```php
<!-- Mostrem la imatge -->
<div class="product-image">
    <img src="imatges/<?php echo htmlspecialchars(string: $p['imatge']); ?>"
    alt="<?php echo htmlspecialchars(string: $p['nom']); ?>">
</div>

<!-- Mostrem el nom -->
<h3><?php echo htmlspecialchars(string: $p['nom']); ?></h3>

<!-- Mostrem la categoria -->
<p><strong>Categoria:</strong> <?php echo htmlspecialchars(string: $p['categoria']); ?></p>

<!-- Mostrem la descripció -->
<p><?php echo htmlspecialchars(string: $p['descripcio']); ?></p>

<!-- Mostrem el preu -->
<p>Preu: <?php echo number_format(num: $p['preu'], decimals: 2); ?> €</p>

<!-- Mostrem l'stock visible (stock de la BD menys el que hi ha el cistell) -->
<p>Estoc disponible: <?php echo $estoc_visible; ?></p>
```

I per últim, l'stock visible (stock de la BD menys productes del cistell), el qual si és 0, mostrem que no hi ha estoc, però si n'hi ha 1 o més, definim un formulari on l'usuari pot indicar quina quantitat de productes pot afegir al cistell.

Aquesta entrada que defineix el nombre de productes a afegir té un màxim limitat per l'stock visible

```php
    <!-- Si l'stock visible es major a 0 (és a dir, l'stock a la BD, restant el que hi ha al cistell) -->
    <?php if ($estoc_visible > 0): ?>
        <!-- Mostrem el formulari per afegir al cistell -->
        <form action="processa.php?accio=afegir" method="post"> <!-- Acció és a proccessa.php definint la variable "accio" com "afegir" -->
            <input type="hidden" name="id" value="<?php echo (int)$p['id']; ?>"> <!-- Afegim un camp ocult el qual és per enviar la ID del producte (el client no ho ha d

            <label>
                <!-- Definim la quantitat que podem afegir al cistell amb una sola operació -->
                Quantitat:
                <input type="number"
                    name="quantitat"
                    value="1"
                    min="1"
                    max="<?php echo $estoc_visible; ?>"> <!-- El nombre màxim d'unitats que es poden afegir al cistell és l'stock visible, per evitar afegir més producte
            </label>

            <input type="submit" value="Afegeix al cistell">
        </form>
    <?php else: ?> <!-- En cas que l'stock visible sigui igual a zero, mostrarem producte esgotat, sense la possibilitat d'agefir al cistell -->
        <p><strong>Producte esgotat</strong></p>
    <?php endif; ?>
    </div>
<?php endforeach; ?>
```

En aquest formulari, afegim un botó per afegir al cistell, el qual crida al codi processa.php definint el valor "afegir" a la variable "acció"

```php
<!-- Mostrem el formulari per afegir al cistell -->
<form action="processa.php?accio=afegir" method="post">
    <input type="hidden" name="id" value="<?php echo (i
```

Aquest procés es realitza per cada producte.

Cistell

En aquesta pàgina es mostra tots els elements afegits al cistell, a més del seu subtotal i el total de la compra. També permet realitzar la compra, buidar el cistell o tornar al catàleg per seguir comprant.

Primer, iniciem la sessió per recuperar els elements afegits al cistell de la pàgina anterior, requerim una connexió de lectura a la BD per recuperar els stocks dels productes i el fitxer funcions.php

```php
<?php
// Iniciem la sessió, requerim la connexió a la BD i el fitxer amb les funcions
session_start();
require_once "funcions.php";
require_once "./connexioBD/connexioR.php";
```

A més, inicialitzem el cistell en cas de no haver-ho estat

```php
// Inicialitzar cistell
if (!isset($_SESSION['cistell'])) {
    $_SESSION['cistell'] = array();
}
```

I inicialitzem la variable del total de la compra. La inicialitzem aquí perquè s'utilitza dins d'un bucle i no hi ha cap altre lloc per declarar-la. Com que és un bucle, si la declaréssim dins, el valor del total es reiniciaria a 0 a cada iteració, i volem que el total s'acumuli al llarg de totes les iteracions

```php
$total = 0; // Inicialitzem la variable del preu total
```

A partir d'aquí, començem a definir l'HTML de la pàgina, on definim el títol i un espai per mostrar missatges d'error o èxit. Aquests missatges únicament surten si estan definits, i es defineixen al codi de Gestió del cistell segons el resultat de l'operació realitzada

```php
<h1>Cistell de la compra</h1>

<!-- Missatge de confirmació d'accions -->
<center>
<?php if (isset($_SESSION['missatge_ok'])): // Quan s'enviï un missatge de confirmació?>
    <p style="color: green; font-weight: bold;">
        <?php
            echo htmlspecialchars(string: $_SESSION['missatge_ok']); // Mostrem el missatge
            unset($_SESSION['missatge_ok']); // Eliminem el missatge, així al recarregar la pàgina no tornarà a sortir,
        ?>
    </p>
<?php endif; ?>
</center>
```

El següent pas és mostrar tots els elements del cistell. Primer es verificarà si és o no buit i en cas que tingui elements es començarà per crear les capçaleres de la taula

```php
<?php if (empty($_SESSION['cistell'])): ?> <!-- En cas que el cistell estigui buit -->
    <p>El cistell està buit.</p>
<?php else: ?> <!-- En cas que el cistell tingui algun element -->
    <table border="1" cellpadding="5"> <!-- Crearem una taula per mostrar els productes -->
        <tr> <!-- Capçaleres -->
            <th>Imatge</th>
            <th>Producte</th>
            <th>Preu unitari</th>
            <th>Quantitat</th>
            <th>Subtotal</th>
            <th>Accions</th>
        </tr>
```

I a continuació, mitjançant un foreach que recorrerà tots els elements del cistell, mostrarem els detalls i calcularem els subtotals de cada producte i el total de compra.

Primer, calcularem els preus del producte (amb descompte i sense) mitjançant la funció calcularPreus.

```php
// Calculem el preu dels productes aplicant un possible descompte
$total_producte = calcularPreus(preu_unitari: $db_producte['preu'], quantitat: $quantitat); // És una array
```

Seguidament, al total de compra li sumem el valor retornat de la funció “total_final”, el qual és el preu final, aplicant-hi descompte si escau

```php
$total_compra = $total_compra + $total_producte['total_final']; // Per cada producte, sumem el seu import al total de la compra
```

A continuació necessitem saber l’stock del producte, per poder posar limits per afegir més productes. La teoria és exactament la mateixa realitzada al catàleg: agafar l’stock de la BD i restar-li la quantitat que ja hi ha al cistell

```php
// Obtenim el seu stock
$stmt = $pdo->prepare(query: "SELECT estoc FROM productes WHERE id = :id");
$stmt->bindParam(param: ":id", var: &$id);
$stmt->execute();
$stockBD = (int)$stmt->fetchColumn();

$maxAfegir = $stockBD - $prod['quantitat']; // Stock real (BD - Cistell)
if ($maxAfegir < 0) {
    $maxAfegir = 0;
}
```

Un cop calculat el subtotal del producte, el total acumulatiu de la compra i el stock del producte, mostrarem tots els detalls del producte

```
<!-- Mostrem tots els detalls dels productes -->
<tr>
    <!-- Imatge -->
    <td><img src="imatges/<?php echo htmlspecialchars(string: $prod['imatge']); ?>" width="80"></td>
    <!-- Nom -->
    <td><?php echo htmlspecialchars(string: $prod['nom']); ?></td>
    <!-- Preu -->
    <td><?php echo number_format(num: $prod['preu'],decimals: 2); ?> €</td>
    <!-- Quantitat -->
    <td><?php echo $prod['quantitat']; ?></td>
```

I en últim lloc, el preu. En cas que hi hagi descompte, es mostrarà el preu original tatxat i el preu descomptat amb el % de descompte aplicat

```
<?php if($preusProducte['descompte_percent'] > 0): ?> <!-- Si el percentatge de descompte és major a 0 (s'aplica un descompte) -->
    <span style="text-decoration: line-through;"><?php echo number_format(num: $preusProducte['total_original'],decimals: 2); ?> €</span>
    &rarr;<!-- És el símbol "→" --> <?php echo number_format(num: $preusProducte['total_final'],decimals: 2); ?> € <!-- Imprimim "→" i el
    (<?php echo $preusProducte['descompte_percent']; ?>% desc.) <!-- Mostrem el % de descompte aplicat -->
```

En cas que no hi hagi cap descompte, únicament mostrem el preu original i res més

```
<?php else: ?>
    <?php echo number_format(num: $preusProducte['total_original'],decimals: 2); ?> €
<?php endif; ?>
```

Per cada producte, afegirem la possibilitat d'afegir i retirar una quantitat de productes del cistell i d'eliminar per complet el producte del cistell.

Per això, s'inclouen dues entrades numèriques i tres botons; un per afegir més quantitats al cistell, un l'altre per retirar-ne i un últim per eliminar el producte del cistell

A l'hora d'afegir productes, si l'stock real (calculat anteriorment, stock BD - quantitat cistell) és 0, no mostrarem cap botó i informarem que no queden existències.

En cas que si que en quedin, mostrarem el botó per afegir elements que defineix un mínim d'1 i un màxim igual a l'stock real. Aquest crida al codi processa.php definint el valor "afegirEnCistell" a la variable "acció"

```
<td style="text-align: center; vertical-align: middle;">
    <div style="display: inline-block;">
    <!-- Afegir -->
    <?php if ($maxAfegir > 0): ?>
        <form action="processa.php?accio=afegirEnCistell" method="post" style="margin-bottom: 10px;">
            <input type="hidden" name="id" value="<?php echo $id; ?>">
            <input type="number"
                name="quantitat"
                value="1"
                min="1"
                max="<?php echo $maxAfegir; ?>">
            <input type="submit" value="Afegir" style="color:white; background-color: green;">
        </form>
    <?php else: ?>
        <em>Sense stock</em><br><br>
    <?php endif; ?>
```

El botó per retirar elements crida al codi processa.php definint el valor “retirar” a la variable “accio”

```
<!-- Retirar productes -->
<form action="processa.php?accio=retirar" method="post">
    <input type="hidden" name="id" value="<?php echo $id; ?>">
    <input type="number"
        name="quantitat"
        value="1"
        min="1"
        max="<?php echo $prod['quantitat']; ?>">
    <input type="submit" value="Retira">
</form>
```

I per últim, el botó per eliminar un producte del cistell crida al codi processa.php definint el valor “eliminar” a la variable “accio”

```
<!-- Eliminar un producte del cistell -->
<br>
<form action="processa.php?accio=eliminar" method="post">
    <input type="hidden" name="id" value="<?php echo $id; ?>">
    <input type="submit" value="Eliminar producte">
</form>
```

Aquest procés comentat es fa amb tots els elements dins del cistell.

Un cop mostrats tots els productes del cistell, mostrarem una última fila amb el total de compra

```
<tr> <!-- Mostrem el total de la compra -->
    <td colspan="4"><strong>Total</strong></td>
    <td colspan="2"><strong><?php echo number_format(num: $total,decimals: 2); ?> €</strong></td>
</tr>
```

I definirem el botó de compra, que porta al codi compra.php

```
<br>
<form action="compra.php" method="post">
    <input type="submit" value="💳 Finalitza la compra">
</form>
```

A més d'un botó per buidar el cistell, el qual porta al codi processa.php definint el valor "buidar" a la variable "acció"

```
<center>
<a href="processa.php?accio=buidar">Buida el cistell</a>
```

Acabant amb un botó per tornar el catàleg

```
    <center>
    <a href="processa.php?accio=buidar">Buida el cist


<?php endif; ?>

<?php if (empty($_SESSION['cistell'])):?>
    <center> <!-- En cas que el catàleg estigui buit,
<?php endif; ?>

<a href="cataleg.php">Torna al catàleg</a>
</center>
```

Veiem que tenim 2 etiquetes d'obertura i només 1 de tancada. Això és degut a que si hi ha elements dins del cistell, la primera etiqueta que és dins d'un if s'executa però no volem que s'executi la segona etiqueta d'obertura, ja que genera un salt de línia.

En cas que el cistell estigui buit, la primera etiqueta no s'executarà per estar dins d'un if, però volem que estigui centrat, i per això definim la segona etiqueta únicament si el cistell està buit

Gestió del cistell

A l'hora de gestionar el cistell es poden produir 3 accions:

- Afegir un element al cistell (des de cataleg.php)
- Afegir més elements al cistell (des de cistell.php)
- Eliminar una quantitat d'un sol producte del cistell (des de cistell.php)
- Buidar per complet el cistell (des de cistell.php)

Aquest codi contempla les 3 opcions, i aplica una o una altra segons la variable "accio".

Aquesta variable es defineix en els formularis dels altres codis, quan a l'enllaç s'inclou el valor de la variable.

Per exemple, aquest formulari:

<form action="processa.php?accio=afegir" method="post">

Redirigeix al codi definint la variable accio amb el valor "afegir". Cada formulari o botó que realitza una de les tres accions canvia el valor de la variable accio segons la seva petició.

Per als formularis fem servir els tipus GET per poder passar valors de les variables via URL.

Un cop arriba la variable al fitxer, el primer que fem (a part de inicialitzar la sessió i requerir connexió a la BD) és desar aquest valor a una variable

```php
<?php
// Iniciem la sessió i requerim la connexió a la BD
session_start();
require_once "./connexioBD/connexioRW.php";

// Inicialitzar cistell
if (!isset($_SESSION['cistell'])) {
    $_SESSION['cistell'] = array();
}

$accio = $_GET['accio']; // L'acció la recuperem dels formularis de catàleg.php o cistell.php
```

Per diferenciar les accions a realitzar segons la variable, fem servir un switch amb els 3 casos comentats anteriorment

```php
switch ($accio) {
```

Afegir productes des del catàleg

En el cas d'afegir un element al cistell des del catàleg, arribaran els següents paràmetres: una ID de producte i una quantitat. (Extracte del codi cataleg.php)

```php
<form action="processa.php?accio=afegir" method="post"> <!-- Acció és a p
    <input type="hidden" name="id" value="<?php echo (int)$p['id']; ?>">

    <label>
        <!-- Definim la quantitat que podem afegir al cistell amb una sol
        Quantitat:
        <input type="number"
            name="quantitat"
            value="1"
            min="1"
            max="<?php echo $estoc_visible; ?>"> <!-- El nombre màxim d'u
    </label>

    <input type="submit" value="Afegeix al cistell">
</form>
```

Primer, haurem de verificar que s'ha proporcionat una ID i que sigui un nombre

```php
// Validar ID
if (!isset($_POST['id']) || !is_numeric(value: $_POST['id'])) { // Verifiquem que el producte té una ID i que és numèrica
    $_SESSION['missatge_error'] =
    "ID de producte no vàlid";
    header(header: "Location: cataleg.php");
    exit;
}

$id = (int)$_POST['id']; // Assignem la ID del producte a una variable verificant que és INT
```

Seguidament, haurem de verificar la quantitat. Si és menor o igual a 0 enviarem un missatge d'error i retornarem l'usuari al catàleg, mostrant el missatge d'error

```php
// Definir i validar quantitat de productes a afegir
$quantitat = (int)$_POST['quantitat']; // Agafa la quantitat del producte
if ($quantitat <= 0) { // En cas que sigui 0 o menor, no es processa la quantitat
    $_SESSION['missatge_error'] =
    "Quantitat no vàlida";
    header(header: "Location: cataleg.php");
    exit;
}
```

El següent pas serà recuperar els detalls del producte segons la ID proporcionada. En cas que la ID sigui errònia, no recuperarà cap producte i es redirigirà al catàleg indicant que el producte no existeix

```php
// Obtenir dades del producte
$stmt = $pdo->prepare(
    query: "SELECT nom, preu, estoc, imatge, categoria
    FROM productes
    WHERE id = :id"
);
$stmt->bindParam(param: ":id",var: &$id);
$stmt->execute();
$producte = $stmt->fetch();

if (!$producte) { // En cas que el producte no existeixi
    $_SESSION['missatge_error'] =
    "El producte no existeix";
    header(header: "Location: cataleg.php");
    exit;
}
```

Per últim, cal afegir el producte al cistell o actualitzar la seva quantitat. En cas que ja existeixi el producte al cistell, li afegim la quantitat rebuda pel formulari

```php
// Afegir o actualitzar producte al cistell
if (isset($_SESSION['cistell'][$id])) { // Si el producte està al cistell
    $_SESSION['cistell'][$id]['quantitat'] = $_SESSION['cistell'][$id]['quantitat'] + $quantitat;
```

En cas que el producte no estigui al cistell, l'afegim amb la quantitat rebuda pel formulari

```php
} else { // En cas que el producte no hi sigui al cistell
    $_SESSION['cistell'][$id] = array( // Inserim el producte al cistell
        'nom' => $producte['nom'],
        'preu' => $producte['preu'],
        'quantitat' => $quantitat, // On la quantitat es defineix segons el nombre de productes a introduir al cistell
        'imatge' => $producte['imatge'],
        'categoria' => $producte['categoria']
    );
}
```

Per últim, mostrem el missatge d'èxit que informa quin i quants productes s'han afegit al cistell

```php
$_SESSION['missatge_ok'] =
"S'ha afegit $quantitat {$producte['nom']} al cistell.";
header(header: "Location: cataleg.php");
exit;
```

Afegir productes des del cistell

En cas d'afegir un producte al cistell des del propi cistell també arribarà la ID del producte i la quantitat (Extracte del codi cistell.php)

```php
<!-- Afegir -->
<?php if ($maxAfegir > 0): ?>
    <form action="processa.php?accio=afegirEnCistell" method="post" style="margin-bottom: 10px;">
        <input type="hidden" name="id" value="<?php echo $id; ?>">
        <input type="number"
            name="quantitat"
            value="1"
            min="1"
            max="<?php echo $maxAfegir; ?>">
        <input type="submit" value="Afegir" style="color:white; background-color: green;">
    </form>
<?php else: ?>
    <em>Sense stock</em><br><br>
<?php endif; ?>
```

En aquest cas, el case és molt semblant al de afegir productes des del catàleg, únicament canvia la redirecció. Primer validem la ID i la quantitat

```php
case 'afegirEnCistell': // En cas que l'acció sigui afegir al cistell (cridada en aquest cas des del cistell)

    // Validar ID
    if (!isset($_POST['id']) || !is_numeric(value: $_POST['id'])) { // Verifiquem que el producte té una ID i que és numèrica
        $_SESSION['missatge_error'] =
        "ID de producte no vàlid";
        header(header: "Location: cistell.php");
        exit;
    }

    $id = (int)$_POST['id']; // Assignem la ID del producte a una variable verificant que és INT

    // Definir i validar quantitat de productes a afegir
    $quantitat = (int)$_POST['quantitat']; // Agafa la quantitat del producte
    if ($quantitat <= 0) { // En cas que sigui 0 o menor, no es processa la quantitat
        $_SESSION['missatge_error'] =
        "Quantitat no vàlida";
        header(header: "Location: cistell.php");
        exit;
    }
```

En aquest cas no cal obtenir les dades del producte, ja que no inserirem cap producte al cistell (ja que ja ho estarà) i no farà falta comprovar si existeix, ja que si el veiem existeix i si no, no.

El que si haurem de fer és recuperar el nom del producte des de la sessió pel missatge

```php
$nom_prod = $_SESSION['cistell'][$id]['nom']; // Obtenim el nom del producte (pel missatge de confirmació)
```

Per últim, actualitzem la quantitat del cistell i mostrem el missatge

```php
// Actualitzem la quantitat
$_SESSION['cistell'][$id]['quantitat'] = $_SESSION['cistell'][$id]['quantitat'] + $quantitat;


$_SESSION['missatge_ok'] =
"S'ha afegit $quantitat {$producte['nom']} al cistell.";
header(header: "Location: cistell.php");
exit;
```

A diferència del codi d'afegir un producte al cistell des del catàleg, no comprovem si el producte existeix al cistell, ja que si ho estem fent des del propi cistell, vol dir que ja existeix.

Si no fos així, no es podria afegir més elements si no fos des del catàleg.

Retirar una quantitat de productes

En el cas de retirar un element al cistell, arribaran els següents paràmetres: una ID de producte i una quantitat. (Extracte del codi cistell.php)

```
<!-- Retirar productes -->
<form action="processa.php?accio=retirar" method="post">
    <input type="hidden" name="id" value="<?php echo $id; ?>">
    <input type="number"
        name="quantitat"
        value="1"
        min="1"
        max="<?php echo $prod['quantitat']; ?>">
    <input type="submit" value="Retira">
</form>
```

Al igual que a l'afegir productes, verifiquem les entrades numèriques de l'ID i de la quantitat

```
// Case que funciona com a "Actualitzar" i en arribar a 0 elimina el producte del cistell
case 'retirar': // En cas que l'acció sigui retirar un producte del cistell (cridada sempre desde cistell.php)
    // Validar ID
    if (!isset($_POST['id']) || !is_numeric(value: $_POST['id'])) { // Verifiquem que el producte té una ID i que és numèrica
        $_SESSION['missatge_error'] =
        "ID de producte no vàlid";
        header(header: "Location: cistell.php");
        exit;
    }

    $id = (int)$_POST['id']; // Assignem la ID del producte a una variable verificant que és INT

    // Definir i validar quantitat de productes a retirar
    $quantitat = (int)$_POST['quantitat']; // Agafa la quantitat del producte
    if ($quantitat <= 0) { // En cas que sigui 0 o menor, no es processa la quantitat
        $_SESSION['missatge_error'] =
        "Quantitat no vàlida";
        header(header: "Location: cistell.php");
        exit;
    }
```

En cas que el producte estigui realment al cistell, obtindrem el nom del producte pel missatge de confirmació i restarem la quantitat retirada a la quantitat del cistell

```
if (isset($_SESSION['cistell'][$id])) { // Si el producte realment està en el cistell
    $nom_prod = $_SESSION['cistell'][$id]['nom']; // Obtenim el nom del producte (pel missatge de confirmació)
    $_SESSION['cistell'][$id]['quantitat'] = $_SESSION['cistell'][$id]['quantitat'] - $quantitat; // Restem la
```

En cas que la quantitat arribi a 0, retirem per complet el producte del cistell

```
if ($_SESSION['cistell'][$id]['quantitat'] <= 0) { // En cas que retirem tants productes fins arribar a 0
    unset($_SESSION['cistell'][$id]); // S'elimina el producte del cistell
}
```

I finalment enviem un missatge d'èxit o error, en cas que el producte no existeixi al cistell

```
    $_SESSION['missatge_ok'] = "S'ha retirat $quantitat unitat(s) de $nom_prod del cistell."; // S'envia el missatge
} else { // En cas que el producte realment no està al cistell
    $_SESSION['missatge_error'] = "El producte no està al cistell.";
}

header(header: "Location: cistell.php");
exit;
```

Eliminar un producte del cistell

En el cas d'eliminar per complet un element del cistell des del propi cistell, arribarà únicament un paràmetre: la ID del producte. (Extracte del codi cistell.php)

```
<!-- Eliminar un producte del cistell -->
<br>
<form action="processa.php?accio=eliminar" method="post">
    <input type="hidden" name="id" value="<?php echo $id; ?>">
    <input type="submit" value="Eliminar producte">
</form>
```

A l'igual que amb la resta d'opcions, haurem de verificar la ID i aprofitarem per recuperar el nom del producte des del propi cistell pel missatge de confirmació

```
case 'eliminar':
    // Validar ID
    if (!isset($_POST['id']) || !is_numeric(value: $_POST['id'])) { // Verifiquem que el producte té una ID i que és numèrica
        $_SESSION['missatge_error'] =
        "ID de producte no vàlid";
        header(header: "Location: cistell.php");
        exit;
    }

    $id = (int)$_POST['id']; // Assignem la ID del producte a una variable verificant que és INT
    $nom_prod = $_SESSION['cistell'][$id]['nom']; // Obtenim el nom del producte (pel missatge de confirmació)
```

Per últim, eliminem l'objecte de la sessió i mostrem el missatge

```
unset($_SESSION['cistell'][$id]); // S'elimina el producte del cistell

$_SESSION['missatge_ok'] = "S'ha eliminat el producte $nom_prod del cistell."; // S'envia el missatge
header(header: "Location: cistell.php");
exit;
```

Buidar el cistell

En el cas de buidar per complet el cistell, no arribarà cap paràmetre i únicament destruirà la sessió

```php
case 'buidar': // En cas de buidar el cistell
    // Destrueix la sessió
    $_SESSION['cistell'] = array();
    session_destroy();
    header(header: "Location: cistell.php");
    exit;
```

Compra

Dins del cistell, trobem un botó amb la finalitat de realitzar la compra: restar l'stock dels productes a la BD i registrar una compra a un historial

El primer que es fa per realitzar la compra es iniciar la sessió, requerir la connexió a la BD i el codi funcions.php i verificar que el cistell es troba amb algun element

```php
<?php
// Iniciem la sessió i requerim la connexió a la BD
session_start();
require_once "./connexioBD/connexioRW.php";

// Requerim un fitxer on definim la funció per calcular el descompte
require_once "funcions.php";

if (!isset($_SESSION['cistell']) || empty($_SESSION['cistell'])) { // En cas que s'accedeixi a la pàgina amb el cistell buit
    header(header: "Location: cataleg.php");
    exit;
}
```

Seguidament es comença una transacció que durarà tot el codi. Això és així ja que en cas d'haver-hi algun error és imperatiu tenir la possibilitat de fer un rollback.

```php
$pdo->beginTransaction();
```

El següent pas serà inicialitzar la variable del total de compra i una array de detalls de productes que més endavant es descriurà degudament.

```php
$total_compra = 0; // Definim el total de la compra
$detalls_compra = []; // Definim una array on especificarem els detalls de cada compra (quantitat, preu, etc)
```

Seguidament, de forma molt similar al cistell, per cada producte mitjançant un foreach verificarem la quantitat de productes major a 0

```php
foreach ($_SESSION['cistell'] as $id_producte => $producte) { // Per cada producte
    $quantitat = (int)$producte['quantitat']; // Agafem la quantitat
    if ($quantitat <= 0){ // En cas que la quantitat sigui 0 o menys
        throw new Exception(message: "Quantitat no vàlida"); // Llençem una excepció per invocar el rollback
    }
```

Després, a partir de la ID del producte del cistell, recuperarem les seves dades i verificarem que el producte realment sigui correcte

```php
// Recuperem el preu i l'stock del producte del cistell
$stmt = $pdo->prepare(query: "SELECT preu, estoc FROM productes WHERE id = :id_producte");
$stmt->bindParam(param: ":id_producte",var: &$id_producte);
$stmt->execute();
$db_producte = $stmt->fetch();

// Si la consulta no retorna res, voldrà dir que l'ID no existeix
if (!$db_producte){
    throw new Exception(message: "El producte no existeix");
}
```

Si, tot i havent-hi verificacions d'stock real i visible, es dóna el cas que al cistell hi ha més productes que stock real, llençarem una excepció

```
// Si per qualsevol motiu, la quantitat del cistell és superior a l'stock, llençarem una excepció
if ($quantitat > $db_producte['estoc']){
    throw new Exception(message: "No hi ha prou estoc del producte ID $id_producte");
}
```

El següent pas serà fer el mateix càlcul de total i subtotal que al cistell

```
// Calculem el preu dels productes aplicant un possible descompte
$total_producte = calcularPreus(preu_unitari: $db_producte['preu'], quantitat: $quantitat);
$total_compra = $total_compra + $total_producte['total_final']; // Per cada producte, sumem
```

Per últim, crearem una array de detalls dels productes. Dins d'aquesta array es guardarà de cada producte:

- ID
- Quantitat
- Preu unitari
- Total final

Per després registrar-ho a la taula compres_detall

```
// Desem dels detalls de la compra
$detalls_compra[] = [ //[] a una array indica afegir al final un objecte (en aquest cas una altra array).
    'id_producte' => $id_producte,
    'quantitat' => $quantitat,
    'preu_unitari' => $db_producte['preu'],
    'total_final' => $total_producte['total_final']
];
 Tot aquest procés es realitza per cada producte dins del cistell
```

Tot aquest procés comentat es farà per cada producte dins del cistell.

Un cop recuperats tots els productes, inserirem a la taula compres el total de la compra i la data de compra. És important recuperar la ID d'aquesta compra per després inserir-la als registres de compres_detall (ja que recordem que és una clau forània)

```
// Registrar la compra
$stmt = $pdo->prepare(query: "INSERT INTO compres (data_compra, total) VALUES (NOW(), :totalCompra)");
$stmt->bindParam(param: ":totalCompra",var: &$total_compra);
$stmt->execute();
$compra_id = $pdo->lastInsertId(); // Desem la ID per relacionar els detalls de la compra a aquesta compra
```

A continuació, a partir de l'array de detalls de compres, per cada producte comprat, inserirem els seus detalls (quantitat comprada, preus, etc) a la taula compres_detall, i la ID de compra la recuperem de l'anterior registre a la taula compres

```php
// Registra els detalls de la compra i actualitzar l'stock dels productes
foreach ($detalls_compra as $detall) { // (Per cada producte comprat) Per cada detall de la compra, el qual és un tipus de producte
    $stmt = $pdo->prepare( // Insertarem els detalls a una base de dades
        query: "INSERT INTO compres_detall
        (compra_id, producte_id, quantitat, preu_unitari, total_producte)
        VALUES (:compra_id, :producte_id, :quantitat, :preu_unitari, :total_producte)"
    );
    $stmt->bindParam(param: ":compra_id", var: &$compra_id); // Relacionem els detalls de la compra a la compra
    $stmt->bindParam(param: ":producte_id", var: &$detall['id_producte']); // Definim quin producte és
    $stmt->bindParam(param: ":quantitat", var: &$detall['quantitat']); // La quantitat
    $stmt->bindParam(param: ":preu_unitari", var: &$detall['preu_unitari']); // El preu unitari
    $stmt->bindParam(param: ":total_producte", var: &$detall['total_final']); // El total d'aquell conjunt d'unitats (incloent-hi des

    $stmt->execute();
```

Per últim, haurem d'actualitzar l'stock de la taula productes segons els productes comprats

```php
// Actualitzem l'stock del producte
$stmt = $pdo->prepare(query: "UPDATE productes SET estoc = estoc - :quantitat WHERE id = :id_producte");
$stmt->bindParam(param: ":quantitat",var: &$detall['quantitat']);
$stmt->bindParam(param: ":id_producte",var: &$detall['id_producte']);
$stmt->execute();
```

Aquest procés el realitzarem tantes vegades com productes comprats.

Per últim, confirmem la transacció i destruim la sessió

```php
$pdo->commit(); // Confirmem la transacció
// Un cop comprats els productes, tanquem la sessió
$_SESSION['cistell'] = array();
session_destroy();
```

Recordem que en cas d'haver-hi algun error, el catch saltarà i es farà un rollback

```php
} catch (Exception $e) {
    if ($pdo->inTransaction()){ // En cas d'haver-hi qualsevol excepció en la transacció
        $pdo->rollBack(); // Fem un rollback per no inserir dades a mitges o errònies
    }
    die("Error en la compra: " . htmlspecialchars(string: $e->getMessage()));
}
?>
```

Un cop realitzada la compra, es mostrarà un missatge de confirmació i un botó per tornar al catàleg

Gestió del catàleg

L'aplicació permet gestionar 3 aspectes del catàleg:

- Visualització d'un històric de compres
- Afegir o retirar stock manualment dels productes
- Afegir nous productes al catàleg

A aquesta administració es pot accedir mitjançant un botó dins del catàleg anomenat "Administració del catàleg"

Tots aquest codis que permeten administrar el catàleg estan protegits amb usuari i contrasenya, els quals són:

- usuari: admin
- contrasenya: cataleg

Per tal d'autoritzar l'accés únicament als responsables del manteniment de l'aplicació web

Històric de compres

L'administració del catàleg té la possibilitat de consultar un històric de compres.

Per això, es recuperen les dades de les taules compres i compres_detall

```php
// Recuperació de l'històric de compres a partir de diferents taules de la BD
try {
    // Obtenim totes les compres amb els seus detalls i dades del producte
    // Fem JOIN per tenir-ho tot en una sola consulta i relacionar noms amb IDs, compres i detalls de compres
    $sql = "
        SELECT
            c.id AS compra_id,
            c.data_compra,
            c.total AS total_compra,
            cd.producte_id,
            cd.quantitat,
            cd.preu_unitari,
            cd.total_producte,
            p.nom AS nom_producte,
            p.categoria
        FROM compres c
        INNER JOIN compres_detall cd ON c.id = cd.compra_id
        INNER JOIN productes p ON cd.producte_id = p.id
        ORDER BY c.data_compra DESC, c.id DESC
    ";

    $stmt = $pdo->prepare($sql);
    $stmt->execute();
    $resultats = $stmt->fetchAll();

} catch (PDOException $e) {
    die("Error BD: " . htmlspecialchars(string: $e->getMessage()));
}
?>
```

Els resultats retornats per la consulta són els detalls de les compres i no les compres en sí. És possible que tinguem 9 resultats però realment son 3 compres, és per això que hem d'agrupar aquests 9 resultats en compres.

Primer, definirem l'array $compres, que serà l'array que contindrà totes les compres i els seus detalls

```php
// Agrupem resultats per compra. Cada fila no és una compra, és un producte d'una compra
$compres = array(); // Definim un array per desar les compres i dins tots els productes
```

Seguidament, per cada compra (cada registre) realitzarem el següent

1. Recuperem la ID de compra del producte
2. (Opcional) En cas que aquesta ID no existeixi dins de l'array $compres
    1. Afegim a l'array $compres un vector que té el valor de la ID de compra, el qual serà una altra array
    2. Dins de l'array de la ID de compra afegirem
        1. La data de compra
        2. El total de la compra
        3. Una altra array "productes" que servirà per emmagatzemar els detalls dels productes comprats
3. Finalment, dins de l'array "productes" inserim les dades la compra (el registre, la fila retornada per SQL, que contempla els detalls del producte

```php
foreach ($resultats as $fila) { // Cada fila és un producte. Per cada producte
    $id = $fila['compra_id']; // Recuperem la ID de la compra a la que correspon
    if (!isset($compres[$id])) { // Si la compra no existeix a l'array, la creem
        $compres[$id] = array(
            'data' => $fila['data_compra'], // Desem la data de la compra
            'total' => $fila['total_compra'], // El total de la compra
            'productes' => array() // Una array on hi haurà tots els productes i els seus detalls
        );
    }
    $compres[$id]['productes'][] = $fila; // Afegim a la compra, dins de productes, els detalls (la fila) del producte
}
```

Finalment, per cada compra a $compres, mostrarem un "bloc" on:

- Es mostrarà la ID de compra
- Es mostrarà la data i el total de la compra
- Es mostrarà una taula on, a partir de l'array "productes" s'obtindrà tots els seus valors, que son els detalls de cada producte

```
// Mostrem cada compra amb les seves dades generals i el detall de productes
foreach ($compres as $id_compra => $dades): // Per cada compra (on hi és desat la data, el total, i els detalls de cada producte)
?>
    <h2>Compra #<?php echo (int)$id_compra; ?></h2> <!-- Títol per identificar la compra -->
    <center>
    <p>
        <strong>Data:</strong> <?php echo htmlspecialchars(string: $dades['data']); ?><br> <!-- Mostrem la data -->
        <strong>Total compra:</strong> <?php echo number_format(num: $dades['total'], decimals: 2); ?> € <!-- Mostrem el total -->
    </p>
    </center>

    <table> <!-- Creem una taula per mostrar els detalls dels productes. Una fila per cada producte -->
        <thead>
            <tr>
                <th>Producte</th>
                <th>Categoria</th>
                <th>Preu unitari</th>
                <th>Quantitat</th>
                <th>Total producte</th>
            </tr>
        </thead>
        <tbody>
```

```
        <?php foreach ($dades['productes'] as $p): ?> <!-- Per cada producte dins la compra -->
            <tr>
                <td><?php echo htmlspecialchars(string: $p['nom_producte']); ?></td> <!-- Mostrem el nom del producte -->
                <td><?php echo htmlspecialchars(string: $p['categoria']); ?></td> <!-- La categoria -->
                <td><?php echo number_format(num: $p['preu_unitari'], decimals: 2); ?> €</td> <!-- El preu unitari -->
                <td><?php echo (int)$p['quantitat']; ?></td> <!-- La quantitat -->
                <td><?php echo number_format(num: $p['total_producte'], decimals: 2); ?> €</td> <!-- El total (amb descompte inclòs) -->
            </tr>
        <?php endforeach; ?>
        </tbody>
    </table>
<?php endforeach; ?>

</body>
</html>
```

També s'ofereix la possibilitat d'esborrar l'històric de compres mitjançnat un botó

```php
<h1>Històric de Compres</h1>
<form method="POST" action="">
    <button name="borrarCompres" type="submit">
        <?php echo "Esborrar registre de compres" ?> <!-- Botó per esborrar històric de compres -->
    </button>
</form>
```

Quan es prem el botó s'executen dues consultes que eliminen tots els registres de les taules compres i compres_detall

```php
// Borrar tots els registres de la taula
if (isset($_POST['borrarCompres'])) { // En cas de prémer el botó d'esborrar l'historial
    try {
        $stmt = $pdo->prepare("DELETE FROM compres_detall");
        $stmt->execute();
        $stmt = $pdo->prepare("DELETE FROM compres");
        $stmt->execute();
    } catch (PDOException $e) {
        echo "<p>Error al eliminar els registres: " . htmlspecialchars(string: $e->getMessage()) . "</p>";
    }
}
```

Afegir o retirar stock d'un producte

L'administrador del catàleg té la possibilitat d'afegir o retirar stock d'un producte en concret sempre que ho necessiti.

Per començar, es requereix una connexió a la BD

```php
<?php
require_once "../connexioBD/connexioRW.php"; // Connexió a la BBDD
```

Seguidament, recuperem tots els productes de la BD per mostrar-los més endavant en el formulari. Aquesta part es realitza sempre, havent o no respòs el formulari.

```php
// Obtenir productes. Aquesta part sempre es realitza per tal de poder mostrar els productes al formulari
$stmt = $pdo->query(query: "SELECT id, nom, estoc FROM productes");
$productes = $stmt->fetchAll(mode: PDO::FETCH_ASSOC);
```

El següent codi PHP és en cas de respondre el formulari, però abans estudiarem el propi formulari.

Primerament, definim el títol i un espai per la mostra d'un missatge d'èxit o error segons el resultat de l'operació

```php
<body>
    <br><br><br><br><br><br><br> <!-- Salts de línia molt poc elegants pero molt eficients per centrar el contingut -->
    <h1>Gestionar Stock de Productes</h1>
    <?php if (!empty($missatge)): ?> <!-- En cas que s'enviï un missatge un cop enviat el formulari -->
        <center><p style="color:green"><?= htmlspecialchars(string: $missatge) ?></p></center> <!-- Missatge d'èxit -->
    <?php endif; ?>
    <?php if (!empty($missatge_error)): ?> <!-- En cas que s'enviï un missatge un cop enviat el formulari -->
        <center><p style="color:red"><?= htmlspecialchars(string: $missatge_error) ?></p></center> <!-- Missatge d'error -->
    <?php endif; ?>
```

Seguidament, mostrarem el formulari, on tindrem un desplegable amb tots els productes. Dins d'aquest desplegable, mitjançant un foreach de la consulta realitzada anteriorment, assignem la ID del producte al valor de l'opció del desplegable i mostrem el nom del producte i el stock actual

```php
<form method="post"> <!-- Formulari per afegir stock -->
    <label for="producte_id">Selecciona un producte:</label>
    <select name="producte_id" id="producte_id" required>
        <?php foreach ($productes as $prod): ?> <!-- Per cada producte del catàleg -->
            <option value="<?= $prod['id'] ?>"> <!-- Creem una opció que té com a valor la ID del producte-->
                <?= htmlspecialchars(string: $prod['nom']) ?> (Stock actual: <?= $prod['estoc'] ?>) <!-- Mostr
            </option>
        <?php endforeach; ?>
    </select>
```

I definim un cap per introduir un nombre enter i dos botons: afegir o retirar

```php
<label for="quantitat">Quantitat a afegir o retirar:</label>
<input type="number" name="quantitat" id="quantitat" min="1" required> <!-- Definim la quantitat a afegir o retirar -->
<button type="submit" name="afegir">Afegir stock</button>
<button type="submit" name="retirar">Retirar stock</button>
```

Un cop s'envia el formulari, dins del try tenim un condicional que comprova si s'ha enviat el formulari, principalment una ID de producte i una quantitat, paràmetres els quals desem en una variable

```php
// Únicament si s'ha respòs al formulari
if (isset($_POST['producte_id']) && isset($_POST['quantitat'])) { // Comprovem si s'ha enviat el formulari
    $producte_id = $_POST['producte_id']; // Recuperem el ID de producte
    $quantitat = (int)$_POST['quantitat']; // I la quantitat
```

Seguidament, hem de determinar l'objectiu del formulari, afegir o retirar stock. Per això revisem quin botó s'ha premut i quin valor ha enviat (afegir o retirar) i ho desem a la variable "accio"

```php
// En cas d'haver premut el botó afegir, l'acció serà afegir
if (isset($_POST['afegir'])){
    $accio = 'afegir';
}

// En cas d'haver premut el botó retirar, l'acció serà retirar
if (isset($_POST['retirar'])){
    $accio = 'retirar';
}
```

A partir de la ID rebuda, busquem en la consulta realitzada en primer lloc el nom del producte i el seu stock

```php
// Busquem el producte seleccionat
foreach ($productes as $prod) { // Dins de tots els productes del catàleg
    if ($prod['id'] == $producte_id) { // El producte que coincideixi la ID pasada en el formulari
        $nom_producte = $prod['nom']; // Recollim el nom del producte
        $stock_actual = $prod['estoc']; // I el seu stock
        break;
    }
}
```

Un cop recuperades les dades realitzarem l'acció. Abans de determinar què realitzar, comprovarem que la quantitat és major a 0,

```php
if ($quantitat > 0) {
```

En cas que sigui afegir, realitzarem un update a la BD passant la ID del producte i la quantitat especificada en el formulari i mostrarem un missatge d'èxit indicant el producte i el número d'unitats afegides

```php
if ($accio === 'afegir') { // En cas que sigui afegir
    $stmt = $pdo->prepare(query: "UPDATE productes SET estoc = estoc + :quantitat WHERE id = :id"); // Sumem la quantitat i l'stock
    $stmt->bindParam(param: ':quantitat', var: &$quantitat);
    $stmt->bindParam(param: ':id', var: &$producte_id);
    $stmt->execute();
    $missatge = "S'han afegit $quantitat unitats al producte $nom_producte.";
```

En cas que sigui retirar, comprovarem si la quantitat retirada és menor a l'stock actual o si supera. És a dir:

- Si tenim 50 unitats i retirem menys de les que hi ha (30, 10, 45, etc)
- Si tenim 50 unitats i retirem igual o més de les que hi ha (50, 75, 90, etc)

En cas de retirar menys unitats de les que hi ha, restarem la quantitat especificada a l'stock de la BD

```php
} elseif ($accio === 'retirar') { // En cas que sigui retirar
    if ($quantitat < $stock_actual) { // I retirem menys productes dels que hi ha en stock
        $stmt = $pdo->prepare(query: "UPDATE productes SET estoc = estoc - :quantitat WHERE id = :id");
        $stmt->bindParam(param: ':quantitat', var: &$quantitat);
        $stmt->bindParam(param: ':id', var: &$producte_id);
        $stmt->execute();
        $missatge = "S'han retirat $quantitat unitats del producte $nom_producte.";
```

I per últim, en cas de retirar totes o més unitats de les que hi ha (cosa impossible) directament establirem el valor a 0, així evitem registrar números negatius

```php
} else { // Si retirem igual o més productes dels que hi ha en stock
    $stmt = $pdo->prepare(query: "UPDATE productes SET estoc = 0 WHERE id = :id"); // Especifiquem l'stock a 0 per evitar stock negatiu
    $stmt->execute(params: [':id' => $producte_id]);
    $missatge = "Retirades totes les unitats del producte $nom_producte.";
}
```

Un cop feta l'actualització, tornarem a fer la la consulta dels productes per oferir la informació de l'stock actualitzada.

Afegir productes al catàleg

L'administrador del catàleg pot afegir nous productes especificant els següents detalls en el formulari:

- Nom del producte
- Preu
- Descripció
- Estoc
- Categoria
- Imatge

```php
<form method="post" action="" enctype="multipart/form-data"> <!-- Tipus de codificació que permet pujar imatges -->
    <h1>Afegir Producte Nou</h1>
    <?php if (isset($missatge)):?> <!-- En cas que s'enviï un missatge un cop enviat el formulari -->
        <center><p style="color: green; font-weight: bold;"> <?php echo $missatge ?> </p></center> <!-- Missatge d'èxit -->
    <?php endif; ?>

    Nom:<br>
    <input type="text" name="nom" required><br><br>

    Preu (€):<br>
    <input type="number" step="0.01" name="preu" required><br><br>

    Descripció:<br>
    <textarea name="descripcio" rows="3" cols="40"></textarea><br><br>

    Estoc:<br>
    <input type="number" name="estoc" min="0" required><br><br>

    Categoria:<br>
    <input type="text" name="categoria" required><br><br>

    Imatge:<br>
    <input type="file" name="imatge" accept="image/*" required><br><br>

    <input type="submit" value="Afegir Producte">
</form>
```

Un cop enviat el formulari, prèviament havent requerit una connexió a la BD i haver definit la ruta del servidor on es desaran les imatges

```php
<?php
require_once "../connexioBD/connexioRW.php"; // Connexió PDO

$directoriImatges = "../imatges/"; // Carpeta on es desa la imatge
```

Recuperarem el nom de la imatge pujada i definirem la ruta final (directori on es desen les imatges + nom de la imatge)

```php
if ($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_FILES['imatge'])) { // En cas que s'envii el formulari
    // Nom del fitxer final
    $nomImatge = basename(path: $_FILES['imatge']['name']); // Agafem el nom de la imatge
    $destí = $directoriImatges . $nomImatge; // Definim el destí on es desarà la imatge (../imatges/nomImatge)
```

Un cop movem el fitxer al directori i hagi sortit tot correctament, inserirem les dades del nou producte a la BD i mostrarem el missatge d'èxit

```php
// Mou el fitxer pujat al directori
if (move_uploaded_file(from: $_FILES['imatge']['tmp_name'], to: $destí)) { // Desa el fitxer al servidor

    // Inserim les dades del producte a la BD
    $stmt = $pdo->prepare(query: "
        INSERT INTO productes (nom, preu, descripcio, estoc, categoria, imatge)
        VALUES (:nom, :preu, :descripcio, :estoc, :categoria, :imatge)
    ");

    $stmt->bindParam(param: ':nom', var: &$_POST['nom']);
    $stmt->bindParam(param: ':preu', var: &$_POST['preu']);
    $stmt->bindParam(param: ':descripcio', var: &$_POST['descripcio']);
    $stmt->bindParam(param: ':estoc', var: &$_POST['estoc']);
    $stmt->bindParam(param: ':categoria', var: &$_POST['categoria']);
    $stmt->bindParam(param: ':imatge', var: &$nomImatge); // Nom del fitxer a la BD

    $stmt->execute();

    $missatge = "Producte afegit correctament"; // Definim un missatge d'èxit
} else {
    echo "Error en pujar la imatge.";
}
```

Funció per el preu dels productes i aplicar descomptes

Aquesta funció serveix per calcular el preu dels productes segons la quantitat i aplicar un possible descompte

El descompte consisteix en restar un 5% del total dels productes per cada 2 productes comprats, fins a un màxim d'un 20%. La taula següent pot servir de referència

Nº de productes	% de descompte
1	0
2	5
3	5
4	10
5	10
6	15
7	15
8	20
+8	20

La funció rep 2 paràmetres: el preu unitari del producte i la quantitat comprada i retorna:

- Total original (preu_unitari * quantitat)
- Total final (preu_unitari * quantitat - descompte)
- % de descompte aplicat

Primer, divideix de forma entera la quantitat de producte entre 2 amb la funció intdiv i recupera el quocient enter de la divisió (grups de 2). Per cada grup de 2 s'aplica un descompte del 5%

Si es troba 3 grups de 2, significa que aplicarà un 15% de descompte

```php
<?php
1 reference
function calcularPreus($preu_unitari, $quantitat): array { // Funció per calcular el descompte, li arriba el preu del producte i la quant
    // El descompte consisteix que cada 2 productes es descompte un 5% del total de la suma del mateix producte, fins a un màxim del 20%
    // Aquesta funció retorna sempre el preu sense descompte i el preu amb descompte

    $blocs = intdiv(num1: $quantitat, num2: 2); // Divideix la quantitat de productes entre 2. (2 productes = 1. 6 productes = 3)
```

Seguidament, multiplica aquests grups de 2 per 5 per trobar el % fins a 20%. Això ho fa la funció min, la qual retorna el valor més petit

Definim el resultat de la multiplicació i 20. Si el % és menor a 20, s'aplicarà aquell %. Si el % és major a 20, s'aplicarà el 20%

```php
$percentatge = min(value: $blocs * 5, values: 20); // Calcula el %, fins a un màxim de 20 (2 productes = 1 bloc * 5 = 5% )
```

Per últim es calculen els preus amb i sense descompte i es retornen els resultats. Ens interessa retornar el preu sense descompte per desar-ho a la BD

```php
// Preu sense descompte
$total_original = $preu_unitari * $quantitat; // Calculem el preu original (el preu unitari del producte + quantitat)

// Preu amb descompte
$total_final = $total_original * (1 - $percentatge / 100); // Calculem el preu final aplicant-hi el descompte

return [
    'total_original' => $total_original, // Retornem el total original (preu_unitari * quantitat)
    'total_final' => $total_final, // Retornem el preu descomptat  (preu_unitari * quantitat - descompte)
    'descompte_percent' => $percentatge // Retornem el percentatge descomptat
];
```

Pràctica 5.3

Objectiu de la pràctica

L'objectiu de la pràctica és dissenyar un sistema d'autenticació d'usuaris amb BD per enregistrar usuaris i accions i sessions i galetes per mantenir el login actiu amb diferents funcionalitats

S'implementa la capacitat de registrar usuaris indicant principalment el nom d'usuari i una contrasenya, entre altres dades personals.

A partir d'aquest usuari contrasenya, l'usuari pot autenticar-se en el sistema per accedir a la seva pàgina privada. Durant l'autenticació pot escollir si mantenir iniciada la sessió tot i tancar el navegador o no.

Gràcies a les sessions i galetes, es pot mantenir la sessió de l'usuari activa tot i tancar la pestanya o el navegador, depenent del que l'usuari desitgi.

A la pàgina privada, la qual és diferent per cada usuari autenticat, mostra les dades personals de l'usuari.

Per cada acció que l'usuari realitzi (registre, login, auto-login logout etc) es desarà a una taula de la BD i també es mostrarà a l'usuari les últimes 10 accions realitzades.

Per últim, s'implementa la possibilitat de tancar la sessió per haver de tornar a autenticar-se

Com a funcionalitats extra s'ha implementat:

- Límit d'intents d'autenticació
- La recuperació de contrasenya
- Autenticació mitjançant 2FA
- Modificació de perfil d'usuari
- Verificació de correu electrònic
- Nivells d'usuaris i administradors
- Tauler d'administrador, on:
    - Es pot veure estadístiques d'autenticació i ús
    - Es pot restablir la contrasenya d'altres usuaris
    - Es pot desbloquejar comptes bloquejats per excés d'autenticacions errònies
- Un servidor VPS per gestionar l'enviament de correus per les següents funcionalitats:
    - Verificació de correu electrònic
    - Autenticació mitjançant 2FA

Demostració

En el següent vídeo es pot comprovar una prova de funcionament de l'aplicació:
demostracioA5.3.mp4

Estructura de directoris

5.3

```
├── Millores opcionals i com implementar-les.txt
├── administracioBD
│   ├── creacioBDiTaula.php
│   ├── eliminacioBD.php
│   └── veureErrors.txt
├── config.php
├── connexioBD
│   ├── connexioR.php
│   └── connexioRW.php
├── css
│   ├── editarPerfil.css
│   ├── index.css
│   ├── login.css
│   ├── mfa.css
│   ├── privada.css
│   ├── recuperacio.css
│   ├── registre.css
│   ├── taulerAdmin.css
│   └── verificaCorreu.css
├── documentacio.txt
├── editarPerfil.php
├── funcions.php
├── index.php
├── login.php
├── logout.php
├── mfa.php
├── privada.php
├── recuperacio.php
├── registre.php
├── taulerAdmin.php
├── testmail.php
└── verificaCorreu.php
```

Documentació del codi de l'aplicació

Creació de la BD i les seves taules

Aquesta aplicació tindrà una BD anomenada "autenticacio" amb codificació UTF8 amb dos taules:

- Taula per emmagatzemar els usuaris, de nom "usuaris"
  - ID
  - Nom d'usuari (únic)
  - Contrasenya
  - Nom complet
  - Email (únic)
  - Email verificat
  - Telèfon (únic)
  - Ciutat
  - Edat
  - Rol
  - Data de registre
  - Token d'autenticació
  - Número d'intents de login
  - Últim intent de login

```php
// Creem la taula usuari, on s'emmagatzemen els usuaris que podran autenticar-se a l'aplicació
$sql = "CREATE TABLE usuaris (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nom_usuari VARCHAR(50) NOT NULL UNIQUE,
    contrasenya VARCHAR(255) NOT NULL,
    nom_complet VARCHAR(100) NOT NULL,
    email VARCHAR(100) NOT NULL UNIQUE,
    email_verificat ENUM('si', 'no') NOT NULL DEFAULT 'no',
    telefon VARCHAR(20) NOT NULL UNIQUE,
    ciutat VARCHAR(100) NOT NULL,
    edat INT NOT NULL,
    rol ENUM('ed_admin', 'vi_admin', 'usuari') DEFAULT 'usuari',
    data_registre DATETIME DEFAULT CURRENT_TIMESTAMP,
    token_recordar VARCHAR(100),
    intents_login INT DEFAULT 0,
    ultim_intent DATETIME DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;";
$conn->exec(statement: $sql);
```

- Taula per emmagatzemar totes les accions dels usuaris, de nom “activitat”
    - ID
    - ID de l’usuari al que correspon l’acció (de la taula usuaris)
    - Acció
    - Data i hora de l’acció
    - IP del client

```
// Creem la taula on es registrarà l'activitat dels usuaris
$sql = "CREATE TABLE activitat (
    id INT AUTO_INCREMENT PRIMARY KEY,
    usuari_id INT NOT NULL,
    accio VARCHAR(50) NOT NULL,
    data_hora DATETIME DEFAULT CURRENT_TIMESTAMP,
    ip_client VARCHAR(45),
    FOREIGN KEY (usuari_id) REFERENCES usuaris(id) ON DELETE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;";
```

Connexions a la BD

Les dues connexions a la BD es diferencien segons els permisos aplicats anteriorment: de lectura i escriptura i únicament de lectura.

La connexió de lectura la qual únicament s'usa en la pàgina privada dels usuaris, és la següent:

```php
<?php
    $servidor = "127.0.0.1";
    $usuari = "convidat";
    $contrasenya = "benvingut";
    $nomDB = "autenticacio";
    $opcions = array(
    PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION,
    PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,
    PDO::MYSQL_ATTR_INIT_COMMAND => "SET NAMES utf8mb4"
    );

    try {
        $pdo = new PDO(dsn: "mysql:host=$servidor;dbname=$nomDB", username: $usuari, password: $contrasenya, options: $opcions);
    } catch(PDOException $e) {
        echo "No es pot connectar. Motiu: " . $e->getMessage();
    }
?>
```

La connexió de lectura i escriptura, que s'usa pràcticament en la resta de pàgines, és la següent:

```php
<?php
    $servidor = "127.0.0.1";
    $usuari = "iot";
    $contrasenya = "iot";
    $nomDB = "autenticacio";
    $opcions = array(
    PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION,
    PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,
    PDO::MYSQL_ATTR_INIT_COMMAND => "SET NAMES utf8mb4"
    );

    try {
        $pdo = new PDO(dsn: "mysql:host=$servidor;dbname=$nomDB", username: $usuari, password: $contrasenya, options: $opcions);
    } catch(PDOException $e) {
        echo "No es pot connectar. Motiu: " . $e->getMessage();
    }
?>
```

Pàgina inicial

A la pàgina inicial de l'aplicació trobem principalment dos botons. El primer convida a un usuari existent a iniciar sessió a l'aplicatiu, mentre que l'altre anima a un usuari que encara no és part de l'ecosistema a registrar-se.

En aquesta pàgina no hi ha cap element PHP a destacar, ja que únicament es composa d'elements HTML i CSS.

Registre d'usuaris

El registre d'usuaris consta d'un formulari amb els següents paràmetres, tots obligatoris:

- Nom d'usuari (únic en l'aplicació)
- Contrasenya
- Nom complet
- Email (únic en l'aplicació)
- Telèfon (únic en l'aplicació)
- Ciutat
- Edat

Un cop enviat el formulari, recuperarem la connexió a la BD i l'arxiu amb les funcions

```php
// No necessitem iniciar la sessió ja que no s'utilitza en aquest codi
require 'funcions.php';
require './connexioBD/connexioRW.php';
```

A continuació, inicialitzarem una variable anomenada "error", la qual emmagatzemarà els errors del registre. A més servirà per verificar si les dades són correctes i es pot procedir a l'enregistramen a la BD

```php
$error = '';
```

Seguidament, recuperarem tots els valors del formulari i retirarem els possibles espais inicials o finals, excepte per la contrasenya i el telèfon

```php
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $nom_usuari = trim(string: $_POST['nom_usuari']);
    $contrasenya = $_POST['contrasenya'];
    $nom_complet = trim(string: $_POST['nom_complet']);
    $email = trim(string: $_POST['email']);
    $telefon = trim(string: $_POST['telefon']);
    $ciutat = trim(string: $_POST['ciutat']);
    $edat = $_POST['edat'];
```

Seguidament, realitzarem les següents verificacions:

- Que tots els camps tinguin un valor definit
- Que l'email sigui vàlid
- Que el telèfon sigui vàlid
- Que la ciutat sigui vàlida
- Que la edat sigui vàlida

En cas que alguna d'aquestes verificacions no sigui exitosa, es registrarà a la variable $error

```php
// Verifiquem que cap camp estigui buit
if (empty($nom_usuari) || empty($contrasenya) || empty($nom_complet) ||
    $error = "Manca algun camp";
}

// Verifiquem que l'email sigui realment un email
if (!filter_var(value: $email, filter: FILTER_VALIDATE_EMAIL)) {
    $error = "L'email introduït no és vàlid";
}

// Verifiquem que el telèfon sigui vàlid
if (strlen(string: $telefon) > 9) {
    $error = "Telèfon no vàlid";
}

// Verifiquem que la ciutat sigui vàlida
if (strlen(string: $ciutat) > 100) {
    $error = "Ciutat no vàlida";
}

// Verifiquem que l'edat sigui vàlida
if ($edat < 18 || $edat > 120) {
    $error = "Edat no vàlida";
}
```

En cas que no hagi succeit cap error i per tant la variable $error no té cap valor, inserirem les dades del nou usuari a la BD

Primerament, mitjançant una consulta SQL comprovarem que no hi hagi cap usuari amb el mateix nom d'usuari, email o telèfon

```php
if ($error === ''){
    try {
        // Comprobar si l'usuari, email o telèfon ja existeixin
        $stmt = $pdo->prepare(query: "SELECT * FROM usuaris WHERE nom_usuari = ? OR email = ? OR telefon = ?");
        $stmt->execute(params: [$nom_usuari, $email, $telefon]);
        if ($stmt->fetch()) {
            $error = "El nom d'usuari, email o telèfon ja existeix.";
```

En cas contrari, inserirem les dades a la BD

```php
// Insertar usuari amb contrasenya encriptada
$hash = encriptar_contrasenya(contrasenya: $contrasenya);
$stmt = $pdo->prepare(query: "
    INSERT INTO usuaris
    (nom_usuari, contrasenya, nom_complet, email, telefon, ciutat, edat)
    VALUES (?, ?, ?, ?, ?, ?, ?)
");

$stmt->execute(params: [
    $nom_usuari,
    $hash,
    $nom_complet,
    $email,
    $telefon,
    $ciutat,
    $edat
]);
```

Un cop inserides, registrarem una activitat amb nom “registre” a la BD amb la funció registrar_activitat

```php
// Registrar activitat
$usuari_id = $pdo->lastInsertId();
registrar_activitat(pdo: $pdo, usuari_id: $usuari_id, accio: 'registre');
```

I redirigirem l’usuari a la pàgina d’inici de sessió

```php
// Redirigir a login
header(header: 'Location: login.php');
exit;
```

En cas que hagi succeït cap error, es mostrarà a la part superior del formulari

```php
<?php if ($error): ?>
    <div class="alert alert-error"><?= htmlspecialchars(string: $error) ?></div>
<?php endif; ?>
```

Inici de sessió

En l'inici de sessió, l'usuari trobarà un formulari per introduir el seu usuari i contrasenya.

A més, en cas que no s'hagi registrat, tindrà un botó per registrar-se. A part, si no recorda de les seves credencials, tindrà un botó per poder recuperar la contrasenya.

Per últim, també se li dona a l'usuari la oportunitat (sense obligar-lo) a autenticar-se amb el 2FA.

A l'inici del codi requerirem iniciar la sessió, la connexió a la BD i les el fitxer amb les funcions

```php
session_start();
require 'funcions.php';
require './connexioBD/connexioRW.php';
```

En cas que l'usuari ja estigui autenticat, es redirigirà a la seva pàgina privada.

```php
// Redirigir si ja hi ha una sessió activa
if (esta_autenticat()) {
    header(header: 'Location: privada.php');
    exit;
}
```

A l'hora d'autenticar-se, hi ha un límit de 3 intents erronis abans de que l'usuari es bloquegi, i en cas de fer-ho ho estarà 2 minuts

```php
// Variables de bloqueig
$max_intents = 3;       // Màxim d'intents erronis
$bloqueig_minuts = 2;  // Temps de bloqueig (minuts)
```

Al respondre el formulari recollirem els valors d'usuari i contrasenya i si ha marcat o no la casella de "recordar-me"

```php
try{
    // LOGIN manual
    if ($_SERVER['REQUEST_METHOD'] === 'POST') {
        $nom_usuari = trim(string: $_POST['nom_usuari']);
        $contrasenya = $_POST['contrasenya'];
        $recordar = isset($_POST['recordar']);
```

Seguidament, realitzarem una consulta SQL per comprovar si existeix l'usuari especificat

```php
$stmt = $pdo->prepare(query: "SELECT * FROM usuaris WHERE nom_usuari = ?");
$stmt->execute(params: [$nom_usuari]);
$usuari = $stmt->fetch();

if ($usuari) {
```

En cas que existeixi, primerament comprovarem si el compte està bloquejat. Per això comprovarem:

- Que el nombre d'intents d'inici de sessió que s'ha enregistrat a la BD sigui major o igual a $max_intents, en aquest cas 3 intents
- Si l'últim intent d'inici de sessió s'ha produit fa menys de $bloqueig_minuts, en aquest cas 2 minuts

Si es cumpleixen els 2 requisits, el compte estarà bloquejat.

```php
// Comprovar si hi ha bloqueig
if ($usuari['intents_login'] >= $max_intents
    && strtotime(datetime: $usuari['ultim_intent']) > strtotime(datetime: "-$bloqueig_minuts minutes"))
```

Per fer el càlcul de les dates fem servir la funció interna de PHP strtotime(), la qual retorna la data introduida en temps UNIX (nº de segons des de 1/1/1970).

strtotime($usuari['ultim_intent']) retorna el número de segons que han passat del 1/1/1970 a la data especificada

strtotime("-$bloqueig_minuts minutes") retorna el número de segons que han passat des de l'1/1/1970 fins fa $bloqueig_minuts minuts

Així, tenim la mateixa "base temporal" amb la que fer càlculs

En cas que el compte estigui bloquejat, mostrarem a l'usuari el temps que queda per desbloquejar-lo.

Per això:

1. Calculem el temps (l'hora) UNIX de l'últim intent
    a. Exemple (Exemple: 15:00 emmagatzemat en format UNIX)
2. Calculem el temps (l'hora) UNIX per que s'acabi el bloqueig
    a. Agafem el temps de l'últim intent i li sumem X minuts en segons, en aquest cas 2 minuts per 60 segons
    b. Exemple: (Exemple: 15:02, emmagatzemat en format UNIX)
3. Calculem els segons restants
    a. Agafem el temps (hora) perquè s'acabi el bloqueig i restem a l'hora actual
    b. Exemple: 15:02 - 15:01 = 00:01 (60 segons)
4. Convertim els segons restants a minuts i segons
    a. 80 segons = 1 minut i 20 segons
5. Mostrem els minuts i segons restants

```php
// Comprovar si hi ha bloqueig
if ($usuari['intents_login'] >= $max_intents
    && strtotime(datetime: $usuari['ultim_intent']) > strtotime(datetime: "-$bloqueig_minuts minutes")) {
    // Calcular tiempo restante
    $ultim_intent_ts = strtotime(datetime: $usuari['ultim_intent']);
    $final_bloqueig_ts = $ultim_intent_ts + ($bloqueig_minuts * 60);
    $segons_restants = $final_bloqueig_ts - time();

    $minuts_restants = floor(num: $segons_restants / 60);
    $segons_restants = $segons_restants % 60;

    $error = "Compte bloquejat temporalment. Torna-ho a provar en {$minuts_restants} minuts i {$segons_restants} segons.";
```

Per tant el fluxe del codi seria el següent:

1. **L'usuari s'equivoca 3 vegades**
2. **Intenta accedir una 4ta, no pot** (3 intents_login, 3 segons últim intent)
3. **Espera 2 minuts** (3 intents_login, 2 minuts últim intent, pot accedir)
4. **Es torna a equivocar una 5na vegada** (3 intents_login, 3 segons últim intent)
5. **Torna a esperar 2 minuts**
6. Accedeix correctament

Un cop comprovat que l'usuari no està bloquejat, verificarem si la contrasenya coincideix amb la de l'usuari amb la funció verificar_contrasenya

```
if (verificar_contrasenya(contrasenya: $contrasenya, hash: $usuari['contrasenya'])) {
```

En cas afirmatiu, actualitzarem el nombre d'intents a 0

```
// Login correcte, resetajer intents
$stmt = $pdo->prepare(query: "UPDATE usuaris SET intents_login = 0, ultim_intent = NULL WHERE id = ?");
$stmt->execute(params: [$usuari['id']]);
```

Seguidament, regenerarem la ID de sessió per evitar atacs de sessió fixada

```
session_regenerate_id(delete_old_session: true); // Protecció session fixation
```

Per acabar creant un vector de nom "usuari" dins de la sessió amb els següents vectors:

- ID de l'usuari dins la BD
- Nom d'usuari
- Nom complet
- Rol, per definir si és administrador o no i quin tipus
- Autenticat, per definir que està autenticat

```
$_SESSION['usuari'] = [
    'id' => $usuari['id'],
    'nom_usuari' => $usuari['nom_usuari'],
    'nom_complet' => $usuari['nom_complet'],
    'rol'         => $usuari['rol'],
    'autenticat' => true
];
```

Si l'usuari ha decidit prémer el botó "Recordar-me", generarem un token únic, el qual emmagatzemarem tant a la BD com a una galeta. També desarem la ID de l'usuari en una altra galeta

```
if ($recordar) {
    // Generar un token únic i desar-lo a la BD
    $token = bin2hex(string: random_bytes(length: 16));
    setcookie(name: 'recordar_id', value: $usuari['id'], expires_or_options: time() + 30*24*60*60, path: '/');
    setcookie(name: 'recordar_token', value: $token, expires_or_options: time() + 30*24*60*60, path: '/');

    $stmt = $pdo->prepare(query: "UPDATE usuaris SET token_recordar = ? WHERE id = ?");
    $stmt->execute(params: [$token, $usuari['id']]);
}
```

En cas que la contrasenya no sigui correcte, actualitzarem el registre de l'usuari a la BD i sumarem els intents d'inici de sessió en 1.

A més, registrarem una acció a la taula activitat amb el nom "error-login".

```php
} else {
    // Si la contrasenya és incorrecte, sumem un intent de login
    $stmt = $pdo->prepare(query: "UPDATE usuaris SET intents_login = intents_login + 1, ultim_intent = NOW() WHERE id = ?");
    $stmt->execute(params: [$usuari['id']]);

    // Registrar intent fallit
    registrar_activitat(pdo: $pdo, usuari_id: $usuari['id'], accio: 'error-login');
    $error = "Usuari o contrasenya incorrecte";
}
```

Però, si pel contrari, hem especificat que ens recordi, quan tanquem el navegador i tornem a entrar a l'aplicació, tindrem les dues galetes definides i entrarem en el següent condicional

```
// AUTO-LOGIN con cookies si no hay sesión activa
if (!esta_autenticat() && isset($_COOKIE['recordar_id'], $_COOKIE['recordar_token'])) {
```

El qual, recuperarà la ID de l'usuari i el token a partir de les galetes

```
$id = $_COOKIE['recordar_id'];
$token = $_COOKIE['recordar_token'];
```

A partir d'aquí, es realitzarà una consulta SQL on es comprovarà quin usuari té aquesta mateixa ID i aquest mateix token

```
// Validar cookies amb la BD
$stmt = $pdo->prepare(query: "SELECT * FROM usuaris WHERE id = ? AND token_recordar = ?");
$stmt->execute(params: [$id, $token]);
$usuari = $stmt->fetch();
```

Si l'usuari existeix (vol dir que la ID i el token són correctes), regenerarem la ID de sessió per evitar atacs de sessió fixada, definirem els mateixos vectors que s'ha comentat anteriorment, registrarem una acció "auto-login" i redirigirem a l'usuari a la seva pàgina privada

```
if ($usuari) {
    session_regenerate_id(delete_old_session: true);
    $_SESSION['usuari'] = [
        'id' => $usuari['id'],
        'nom_usuari' => $usuari['nom_usuari'],
        'nom_complet' => $usuari['nom_complet'],
        'rol'        => $usuari['rol'],
        'autenticat' => true
    ];
    registrar_activitat(pdo: $pdo, usuari_id: $usuari['id'], accio: 'auto-login');
    header(header: 'Location: privada.php');
    exit;
```

En cas que la consulta no trobi cap usuari, voldrà dir que o l'ID d'usuari és incorrecte, o el token que està provant de comprovar no és el mateix que el que hi ha a la BD i per tant no s'ha realitzat el procediment de forma correcta i s'ha vulnerat la integritat de les dades.

Per això, si comprovem qualsevol incongruència, esborrarem les galetes (en teoria, amb valors erronis) per evitar noves comprovacions innecessàries.

```
} else {
    // Cookie inválida -> borrar
    setcookie(name: 'recordar_id', value: '', expires_or_options: time() - 3600, path: '/');
    setcookie(name: 'recordar_token', value: '', expires_or_options: time() - 3600, path: '/');
}
```

En cas que hagi succeït cap error, es mostrarà a la part superior del formulari

```php
<?php if ($error): ?>
    <div class="error-container">
        <p style='color:red;'><?= $error ?></p>
    </div>
<?php endif; ?>
```

En aquest codi no hi ha missatges de confirmació ja que si les credencials són correctes, directament redirigeix a la pàgina privada de l'usuari

Inici de sessió amb 2FA

L'usuari té la possibilitat de iniciar sessió mitjançant 2FA. El segon factor d'autenticació és el correu electrònic que ha registrat. El procés consisteix en enviar un codi de 8 dígits al correu indicat al registre i l'usuari ha d'entrar aquest codi en l'inici de sessió.

És molt important tenir en compte que per poder autenticar-se amb 2FA, cal haver verificat el correu electrònic abans.

Primerament, iniciarem la sessió i requerirem la connexió a la BD i el fitxer amb les funcions

```php
<?php
date_default_timezone_set(timezoneId: 'Europe/Madrid');
session_start();
require 'funcions.php';
require './connexioBD/connexioRW.php';
```

Aquest codi es divideix en 2 passos. El primer és mostrar el mateix formulari d'inici de sessió i el segon és mostrar el formulari per introduir el codi de verificació. Per determinar en quin pas està fem servir la variable "pas"

```php
$pas = 1; // Pas 1: Login, Pas 2: Codi MFA
```

.

Depenent del valor de la variable $pas es mostrarà un HTML o un altre

```php
<?php if ($pas == 1): ?>
    <p class="info-text">
```

```php
<?php if ($pas == 1): ?>
    <p class="info-text">Introdueix les teves credencials per rebre el codi per correu.</p>
    <?php if ($error): ?>
        <div class="error-container">
            <p style="color:red;"><?= $error ?></p>
        </div>
    <?php endif; ?>
    <form method="post">
        <div class="input-group">
            <label>Nom d'usuari:</label>
            <input type="text" name="nom_usuari" required>
        </div>
        <div class="input-group">
            <label>Contrasenya:</label>
            <input type="password" name="contrasenya" required>
        </div>
        <div class="checkbox-group">
            <input type="checkbox" name="recordar" id="recordar">
            <label for="recordar">Recordar-me</label>
        </div>
        <input type="submit" value="Enviar codi de seguretat" class="btn-2fa">
    </form>
<?php else: ?>
    <p class="info-text">T'hem enviat un codi de 8 dígits al teu correu.</p>
    <form method="post">
        <div class="input-group">
            <label>Codi MFA:</label>
            <input type="text" name="codi_mfa" placeholder="00000000" required autofocus>
        </div>
        <input type="submit" value="Verificar i Entrar" class="btn-2fa">
    </form>
<?php endif; ?>
```

Pas 1: Login

Pas 2: Codi 2FA

En cas que ja estigui autenticat, redirigirem l'usuari a la pàgina privada

```php
// Si ja està autenticat, fora
if (esta_autenticat()) {
    header(header: 'Location: privada.php');
    exit;
}
```

El codi del formulari és exactament el mateix que l'inici de sessió normal. Compta amb verificació d'usuari i contrasenya i límit d'autenticacions, amb la diferència de que si les credencials són correctes, enlloc de crear una sessió, realitzarem altres comprovacions

```php
if ($_SERVER['REQUEST_METHOD'] === 'POST') {

    // --- FASE 1: VALIDACIÓ DE CREDENCIALS ---
    if (isset($_POST['nom_usuari']) && isset($_POST['contrasenya']) && !isset($_POST['codi_mfa'])) {
        $nom_usuari = trim(string: $_POST['nom_usuari']);
        $contrasenya = $_POST['contrasenya'];
        $recordar = isset($_POST['recordar']);

        $stmt = $pdo->prepare(query: "SELECT * FROM usuaris WHERE nom_usuari = ?");
        $stmt->execute(params: [$nom_usuari]);
        $usuari = $stmt->fetch();

        if ($usuari) {
            // 1. Comprovar si l'usuari està bloquejat temporalment
            if ($usuari['intents_login'] >= $max_intents
                && strtotime(datetime: $usuari['ultim_intent']) > strtotime(datetime: "-$bloqueig_minuts minutes")) {

                $ultim_intent_ts = strtotime(datetime: $usuari['ultim_intent']);
                $final_bloqueig_ts = $ultim_intent_ts + ($bloqueig_minuts * 60);
                $segons_restants = $final_bloqueig_ts - time();

                $minuts = floor(num: $segons_restants / 60);
                $segons = $segons_restants % 60;

                $error = "Compte bloquejat temporalment. Torna-ho a provar en {$minuts} min i {$segons} seg.";
            } else {
                // 2. Intentar validar contrasenya
                if (verificar_contrasenya(contrasenya: $contrasenya, hash: $usuari['contrasenya'])) {
```

En cas que les credencials siguin correctes, verificarem que l'usuari hagi verificat el correu electrònic abans. En cas que no ho hagi fet, registrarà un error i no podrà procedir amb l'autenticació.

```php
// Verificació d'email verificat
if ($usuari['email_verificat'] !== 'si') {
    $error = "No pots usar 2FA perquè el teu correu no està verificat. Inicia sessió de forma normal i verifica'l.";
```

A continuació, ja que l'inici de sessió a sigut exitós, restablirem el nombre d'intents a 0

```php
// Èxit Fase 1: Reset intents i generar codi
$stmt = $pdo->prepare(query: "UPDATE usuaris SET intents_login = 0, ultim_intent = NULL WHERE id = ?");
$stmt->execute(params: [$usuari['id']]);
```

I generarem un nombre de 8 dígits que serà el codi de verificació

```php
$codi = random_int(min: 10000000, max: 99999999);
```

Seguidament, ja que al canviar de pas hem de refrescar la pàgina, hem de guardar els valors de l'usuari dins de la sessió per no perdre'ls

En concret, guardarem:

- El nom d'usuari
- El codi de verificació
- Si vol o no que l'aplicació recordi l'usuari (galetes)

Seguidament, prepararem les capçaleres del correu:

- Destinatari: l'email de l'usuari
- Assumpte: Codi de verificació
- Missatge: "El codi generat anteriorment"
- Altres capçaleres: per complir amb stàndards de seguretat de Nominalia. Si un correu no té unes bones capçaleres, pot ser interpretat com spam

```php
// Enviar mail
$to = $usuari['email'];
$subject = "Codi de seguretat 2FA";
$message = "Hola {$usuari['nom_complet']},\n\nEl teu codi és: $codi\n\nSalutacions, l'equip de gserrat.cat";
$headers = [
    "From: no-reply@gserrat.cat",
    "Reply-To: no-reply@gserrat.cat",
    "MIME-Version: 1.0",
    "Content-Type: text/plain; charset=utf-8",
    "Date: " . date(format: "r"),
    "X-Mailer: PHP/" . phpversion()
];
```

A continuació, enviarem el correu amb la funció interna de PHP mail(). Aquesta funció espera rebre:

- Destinatari
- Assumpte
- Missatge
- Altres capçaleres

Per tant indicarem les variables anteriorment definides. En cas de les altres capçaleres, degut a que és una array amb diferents vectors, farem servir la funció interna de PHP implode(), la qual ajunta els elements d'una array com cadenes de text amb un separador indicat.

```php
// Enviem l'email fent servir la funció mail() configurada a PHP.ini
if (mail(to: $to, subject: $subject, message: $message, additional_headers: implode(s…"\r\n", $headers))) {
    $pas = 2;
} else {
    $error = "Error enviant el correu.";
}
```

És molt important configurar PHP.ini per indicar a la funció mail quin servidor de correu electrònic local fer servir per l'enviament (Postfix, msmtp, etc). Per més informació sobre com configurar un servidor per l'enviament de correus electrònics es pot consultar l'apartat "Configuració del VPS per l'enviament de correus electrònics amb PHP"

Si el mail s'envia correctament sense cap error, passarem al pas 2 i per tant la condició de l'HTML serà diferent i mostrarà un formulari diferent, el d'introduir el codi de verificació

En aquests moments ens haurà arribat un correu electrònic amb el codi de verificació i en l'aplicació tindrem un formulari per introduir aquest mateix codi.

En el moment que enviem el formulari amb un codi de verificació, el recuperarem i tornarem a especificar el pas nº2, ja que si hi ha un error, l'usuari podrà tornar a intentar introduir el codi.

```php
// --- FASE 2: VALIDACIÓ DEL CODI MFA ---
if (isset($_POST['codi_mfa'])) {
    $codi_introduit = trim(string: $_POST['codi_mfa']);
    $pas = 2; // Mantinguem la vista del codi si hi ha error
```

En cas que el codi introduit sigui correcte, recuperarem el nom d'usuari de la sessió i iniciarem la sessió de l'usuari amb els mateixos vectors que a l'inici de sessió normal

```php
if ($codi_introduit == $_SESSION['mfa_codi']) {
    // ÈXIT TOTAL -> Loguegem a l'usuari
    $usuari = $_SESSION['mfa_temp_user'];

    session_regenerate_id(delete_old_session: true);
    $_SESSION['usuari'] = [
        'id' => $usuari['id'],
        'nom_usuari' => $usuari['nom_usuari'],
        'nom_complet' => $usuari['nom_complet'],
        'rol'         => $usuari['rol'],
        'autenticat' => true
    ];
```

A més, també gestionarem de la mateixa forma el cas que l'usuari vulgui que l'aplicació s'enrecordi d'ell al tancar el navegador

```php
// Gestió de cookies "Recordar-me"
if ($_SESSION['mfa_recordar']) {
    $token = bin2hex(string: random_bytes(length: 16));
    setcookie(name: 'recordar_id', value: $usuari['id'], expires_or_options: time() + 30*24*60*60, path: '/');
    setcookie(name: 'recordar_token', value: $token, expires_or_options: time() + 30*24*60*60, path: '/');
    $stmt = $pdo->prepare(query: "UPDATE usuaris SET token_recordar = ? WHERE id = ?");
    $stmt->execute(params: [$token, $usuari['id']]);
}
```

Per últim, netejarem totes les dades temporals de la sessió per l'MFA, registrarem una nova acció a la taula activitat de nom "login-2fa" i redirigirem l'usuari a la seva pàgina privada

```php
// Neteja de dades temporals MFA
unset($_SESSION['mfa_temp_user'], $_SESSION['mfa_codi'], $_SESSION['mfa_recordar']);

registrar_activitat(pdo: $pdo, usuari_id: $usuari['id'], accio: 'login-2fa');
header(header: 'Location: privada.php');
exit;
```

En cas que hagi succeït cap error, tant en la verificació de credencials com a la verificació del codi 2FA es mostrarà a la part superior del formulari

```php
<?php if ($error): ?>
    <div class="error-container">
        <p style='color:red;'><?= $error ?></p>
    </div>
<?php endif; ?>
```

En aquest codi no hi ha missatges de confirmació ja que si les credencials són correctes, es mostra el formulari del codi 2FA (on s'informa que s'ha enviat un correu a l'adreça de correu electrònic) i en cas que el codi de verificació també sigui correcte, es redirigirà a la pàgina privada

Recuperació de contrasenya

Per poder recuperar la contrasenya, s'ha d'emplenar un formulari de dades personals, on s'ha d'especificar:

- Nom d'usuari
- Correu electrònic
- Telèfon
- Edat
- Ciutat
- Nova contrasenya

Per restablir la contrasenya, totes les dades anteriors han de coincidir exactament amb un usuari per verificar la seva identitat.

Per això un cop respòs el formulari, recuperarem la connexió a la BD i el fitxer amb les funcions

```php
<?php
// No necessitem iniciar la sessió ja que no s'utilitza en aquest codi
require 'funcions.php';
require './connexioBD/connexioRW.php';
```

A més, inicialitzarem les variables de missatge i error

```php
$error = '';
$missatge = '';
```

A continuació, recuperarem els valors del formulari

```php
if ($_SERVER['REQUEST_METHOD'] === 'POST') {

    $nom_usuari   = trim(string: $_POST['nom_usuari']);
    $email        = trim(string: $_POST['email']);
    $telefon      = trim(string: $_POST['telefon']);
    $ciutat       = trim(string: $_POST['ciutat']);
    $edat         = $_POST['edat'];
    $nova_pass    = $_POST['nova_contrasenya'];
```

I realitzarem una consulta SQL cercant un usuari amb exactament els mateixos valors en els camps definits

```
try {
    // Busquem un usuari que tingui exactament els mateixos valors
    $stmt = $pdo->prepare(query: "
        SELECT id
        FROM usuaris
        WHERE nom_usuari = ?
          AND email = ?
          AND telefon = ?
          AND ciutat = ?
          AND edat = ?
    ");
    $stmt->execute(params: [$nom_usuari, $email, $telefon, $ciutat, $edat]);

    $usuari = $stmt->fetch(mode: PDO::FETCH_ASSOC);
```

En cas que l'usuari existeixi, actualitzarem la contrasenya. Primerament l'encriptarem mitjançant la funció encriptar_contrasenya.

Seguidament inserirem la nova contrasenya a la BD a partir de la ID recuperada de la consulta anterior i registrarem una nova acció amb nom "recuperacio-contrasenya" mitjaçant la funció registrar_activitat.

Per acabar, definirem un missatge indicant que s'ha actualitzat la contrasenya correctament.

```
// Si l'usuari existeix, vol dir que ha emplenat tots els valors correctament
if ($usuari) {
    // Actualitzem la contrasenya
    $hash = encriptar_contrasenya(contrasenya: $nova_pass);

    $update = $pdo->prepare(query: "
        UPDATE usuaris
        SET contrasenya = ?
        WHERE id = ?
    ");
    $update->execute(params: [$hash, $usuari['id']]);

    // Registrem "recuperacio-contrasenya" a la taula activitat
    registrar_activitat(pdo: $pdo, usuari_id: $usuari['id'], accio: 'recuperacio-contrasenya');

    $missatge = "Contrasenya actualitzada correctament.";
```

En cas que l'usuari no existeixi (ja que no hi ha cap usuari amb el que tots els seus valors coincideixin), informarem de l'error

```
} else {
    $error = "Les dades introduïdes no coincideixen amb cap usuari.";
}
```

En cas que hagi succeït cap error o s'hagi realitzat la operació amb èxit, es mostrarà a la part superior del formulari

```
<?php if ($error): ?>
    <div class="alert alert-error"><?= htmlspecialchars(string: $error) ?></div>
<?php endif; ?>

<?php if ($missatge): ?>
    <div class="alert alert-success"><?= htmlspecialchars(string: $missatge) ?></div>
<?php endif; ?>
```

Pàgina privada

Quan un usuari està autenticat pot accedir a la seva pàgina privada.

En aquesta pàgina es mostra les seves dades personals, historial d'activitat (inicis i tancament de sessió, canvis de contrasenya, inicis de sessió incorrectes, etc) i una mètrica amb estadístiques d'accés (número d'autenticacions, tipus, errors, etc)

En aquesta pàgina es troba un botó per tancar sessió, per modificar el perfil d'usuari i, en cas de ser un usuari administrador, accedir al tauler d'administració.

Per això, iniciarem la sessió per recuperar les dades de l'usuari, com la seva ID, el seu rol o si està o no autenticat. A més recuperarem la connexió a la BD i el fitxer amb les funcions.

```php
<?php
session_start();
require 'funcions.php';
require './connexioBD/connexioRW.php';
```

Degut a que aquesta pàgina és privada i únicament s'hi pot accedir mitjançant un usuari autenticat, hem de verificar que l'usuari estigui autenticat amb la funció requerir_autenticacio.

```php
// Proteger página
requerir_autenticacio();
```

Seguidament, recuperarem l'ID de l'usuari que ha iniciat sessió a partir de la sessió i esborrarem qualsevol codi de verificació residual que hagi quedat de l'autenticació amb 2FA o la verificació del correu electrònic.

```php
// ID de l'usuari
$usuari_id = $_SESSION['usuari']['id'];

unset($_SESSION['codi_verificacio']); // eliminar codi de sessió
```

A continuació recuperarem les dades bàsiques de l'usuari

```php
try {
    // Consultar dades de l'usuari
    $stmt = $pdo->prepare(query: "
        SELECT nom_usuari, nom_complet, email, email_verificat, telefon, ciutat, edat
        FROM usuaris
        WHERE id = ?
    ");
    $stmt->execute(params: [$usuari_id]);
    $usuari = $stmt->fetch();
```

A més dels últims 10 registres de la taula "activitat" del propi usuari i l'últim accés a l'aplicació

```php
// Consultar últimes 10 accions i últim login
$stmt = $pdo->prepare(query: "SELECT accio, ip_client, data_hora FROM activitat WHERE usuari_id = ? ORDER BY data_hora DESC LIMIT 10");
$stmt->execute(params: [$usuari_id]);
$historial = $stmt->fetchAll();

// Últim accés a l'aplicació
$stmt = $pdo->prepare(query: "
    SELECT data_hora
    FROM activitat
    WHERE usuari_id = ?
    AND accio IN ('login', 'auto-login', 'login-2fa')
    ORDER BY data_hora DESC
    LIMIT 1
");
```

I per acabar, les mètriques

```php
// Total autenticacions
$stmt = $pdo->prepare(query: "
    SELECT
        SUM(accio = 'login')      AS logins_manuals,
        SUM(accio = 'auto-login') AS auto_logins,
        SUM(accio = 'login-2fa') AS logins_2fa,
        SUM(accio = 'error-login') AS error_logins,
        SUM(accio = 'restablir-contrasenya') AS restabliments_contrasenya,
        SUM(accio IN ('login', 'auto-login', 'login-2fa')) AS total
    FROM activitat
    WHERE usuari_id = ?
");
$stmt->execute(params: [$usuari_id]);
$autenticacions = $stmt->fetch();
```

Tots aquests resultats els mostrarem a l'HTML, sempre amb escapament per evitar atacs XSS.

Les dades personals

```
<section class="user-data-card full-width">
    <div class="card-header">
        <h3>Les teves dades personals</h3>
        <a href="editarPerfil.php" class="btn-edit">Editar Perfil</a>
    </div>
    <div class="data-grid">
        <div class="data-item"><strong>Nom d'usuari:</strong> <?= htmlspecialchars(string: $usuari['nom_usuari']) ?></div>
        <div class="data-item"><strong>Nom complet:</strong> <?= htmlspecialchars(string: $usuari['nom_complet']) ?></div>
        <div class="data-item"><strong>Ciutat:</strong> <?= htmlspecialchars(string: $usuari['ciutat'] ?: '-') ?></div>
        <div class="data-item"><strong>Telèfon:</strong> <?= htmlspecialchars(string: $usuari['telefon'] ?: '-') ?></div>
        <div class="data-item"><strong>Edat:</strong> <?= htmlspecialchars(string: $usuari['edat'] ?: '-') ?> anys</div>
        <div class="data-item email-status">
            <strong>Email:</strong> <?= htmlspecialchars(string: $usuari['email']) ?>
            <?php if ($usuari['email_verificat'] === 'si'): ?>
                <span class="badge badge-success">Verificat</span>
            <?php else: ?>
                <span class="badge badge-danger">No verificat <a href="verificaCorreu.php">(Verificar ara)</a></span>
            <?php endif; ?>
```

Les últimes accions

```
<section class="card history-card">
    <h3>Historial d'activitat</h3>
    <div class="activity-list">
        <?php foreach ($historial as $index => $act): ?>
            <div class="activity-item <?= $index === 0 ? 'latest' : '' ?>"> <!-- Si és el primer de la llista afegim la classe "latest" -->
                <span class="activity-date"><?= htmlspecialchars(string: $act['data_hora']) ?></span>
                <span class="activity-action"><?= htmlspecialchars(string: $act['accio']) ?></span>
                <span class="activity-ip"><?= htmlspecialchars(string: $act['ip_client']) ?></span>
            </div>
        <?php endforeach; ?>
    </div>
</section>
```

I les estadístiques d'autenticació

```
<section class="card stats-card">
    <h3>Estadístiques d'accés</h3>
    <div class="stats-list">
        <div class="stat-row"><span>Total autenticacions</span> <strong><?= $autenticacions['total'] ?></strong></div>
        <div class="stat-row"><span>Logins manuals</span> <strong><?= $autenticacions['logins_manuals'] ?></strong></div>
        <div class="stat-row"><span>Auto-logins</span> <strong><?= $autenticacions['auto_logins'] ?></strong></div>
        <div class="stat-row"><span>Logins 2FA</span> <strong><?= $autenticacions['logins_2fa'] ?></strong></div>
        <div class="stat-row"><span>Logins erronis</span> <strong class="text-danger"><?= $autenticacions['error_logins'] ?></strong></div>
        <div class="stat-row"><span>Restabliments de contrasenya</span> <strong><?= $autenticacions['restabliments_contrasenya'] ?></strong></div>
    </div>
</section>
```

Com s'ha comentat, en cas de ser un usuari administrador, en la secció de dades personals es trobarà un botó per accedir al tauler d'administració

```
<?php if (es_admin()): ?>
    <div class="admin-zone">
        <a href="taulerAdmin.php" class="btn-admin">Accedir al Tauler d'Administració</a>
    </div>
<?php endif; ?>
```

Edició de perfil d'usuari

En cas que l'usuari destiji modificar les seves dades d'usuari comptarà amb un formulari on no només podrà modificar les seves dades personals sinó també la contrasenya.

Per accedir a aquest formulari és imperatiu que l'usuari estigui autenticat, és per això que requerim la funció requerir_autenticacio

```
// 1. Verificació d'autenticació
requerir_autenticacio();
```

El formulari implementa una funcionalitat que mostra les dades a la BD com a valors predefinits en els inputs, així l'usuari sap quins valors hi havien i pot modificar-los més fàcilment.

També, a nivell de gestió de BD, podrem fer un update de tot sense haver de comprovar quin camp ha canviat l'usuari, fent-ho més senzill

Per això, recuperarem les dades de la sessió i requerirem la connexió a la BD i el fitxer de funcions

```
session_start();
require 'funcions.php';
require './connexioBD/connexioRW.php';
```

A continuació, realitzarem una consulta SQL per recuperar les dades de l'usuari

```php
$error = '';
$correcte = '';

// 2. Obtenir ID de l'usuari autenticat
$usuari_id = $_SESSION['usuari']['id'];

// 3. Obtenir les dades de l'usuari
try {
    $stmt = $pdo->prepare(query: "
    SELECT nom_usuari, nom_complet, email, telefon, ciutat, edat, contrasenya, email_verificat
    FROM usuaris
    WHERE id = ?
    ");
    $stmt->execute(params: [$usuari_id]);
    $usuari = $stmt->fetch();
} catch (PDOException $e) {
    $error = "No s'ha pogut recuperar les dades de l'usuari. Motiu: " . $e->getMessage();
}


if (!$usuari) {
    die("Usuari no trobat");
}
```

I en el formulari especificarem l'atribut "value" dels inputs segons el resultat de la consulta SQL

```php
<div class="input-group">
    <label>Nom d'usuari</label>
    <input type="text" name="nom_usuari" value="<?= htmlspecialchars(string: $usuari['nom_usuari']) ?>" required>
</div>
<div class="input-group">
    <label>Nom complet</label>
    <input type="text" name="nom_complet" value="<?= htmlspecialchars(string: $usuari['nom_complet']) ?>" required>
</div>
<div class="input-group full-width">
    <label>Email <span class="info-tag">(Es requerirà nova verificació si es canvia)</span></label>
    <input type="email" name="email" value="<?= htmlspecialchars(string: $usuari['email']) ?>" required>
</div>
<div class="input-group">
    <label>Telèfon</label>
    <input type="text" name="telefon" value="<?= htmlspecialchars(string: $usuari['telefon']) ?>" required>
</div>
<div class="input-group">
    <label>Ciutat</label>
    <input type="text" name="ciutat" value="<?= htmlspecialchars(string: $usuari['ciutat']) ?>" required>
</div>
<div class="input-group">
    <label>Edat</label>
    <input type="number" name="edat" min="0" max="120" value="<?= htmlspecialchars(string: $usuari['edat']) ?>" required>
</div>
```

En cas de les contrasenyes es mostrarà buit per obvies raons i es comprovarà si l'usuari ha o no especificat algún valor.

Un cop s'ha enviat el formulari, recuperarem els valors d'aquest

```php
// 4. Processar formulari
if ($_SERVER['REQUEST_METHOD'] === 'POST') {

    $nom_usuari  = trim(string: $_POST['nom_usuari']);
    $nom_complet = trim(string: $_POST['nom_complet']);
    $email       = trim(string: $_POST['email']);
    $telefon     = trim(string: $_POST['telefon']);
    $ciutat      = trim(string: $_POST['ciutat']);
    $edat        = trim(string: $_POST['edat']);
    $contrasenya_actual     = $_POST['contrasenya_actual'];
    $nova_contrasenya       = $_POST['nova_contrasenya'];
    $confirmar_contrasenya = $_POST['confirmar_contrasenya'];
```

A continuació, comprovarem si algun camp dels que són únics a l'aplicació ha canviat, ja que si és així, haurem de verificar si ja existeix o no

```php
// Determinar si l'email, el telèfon o el nom d'usuari ha canviat
$email_ha_canviat = ($email !== $usuari['email']);
$telefon_ha_canviat = ($telefon !== $usuari['telefon']);
$usuari_ha_canviat = ($nom_usuari !== $usuari['nom_usuari']);
```

A més, també comprovarem si es vol modificar la contrasenya, verificacant que els tres camps tenen algun valor

```php
// Determinar si se quiere cambiar la contraseña
$canvi_contrasenya = (
    $contrasenya_actual !== '' ||
    $nova_contrasenya !== '' ||
    $confirmar_contrasenya !== ''
);
```

A continuació, farem les validacions bàsiques, les mateixes que a l'hora del registre d'usuaris

```php
// Validacions bàsiques
// Verifiquem que cap camp estigui buit
if ($nom_usuari === '' || $nom_complet === '' || $email === '' || $ciutat === '' || $telefon === '' || $edat === '') {
    $error = "Els camps obligatoris no poden estar buits";
}

// Verifiquem que l'email sigui realment un email
if (!filter_var(value: $email, filter: FILTER_VALIDATE_EMAIL)) {
    $error = "L'email introduït no és vàlid";
}

// Verifiquem que el telèfon sigui vàlid
if (strlen(string: $telefon) > 9) {
    $error = "Telèfon no vàlid";
}

// Verifiquem que la ciutat sigui vàlida
if (strlen(string: $ciutat) > 100) {
    $error = "Ciutat no vàlida";
}

// Verifiquem que l'edat sigui vàlida
if ($edat < 18 || $edat > 120) {
    $error = "Edat no vàlida";
}
```

En cas que no es compleixi algún requisit, es registrarà l'error

A partir d'aquest moment, comprovarem els camps únics en cas que s'hagin modificat.

Per cada camp, comprovarem si hi ha un usuari amb el mateix valor. Únicament es farà la comprovació si realment ha estat modificat, per evitar càrrega a la BD.

Cada camp també comprova si hi ha hagut errors anteriorment, ja només que hi hagi un error ja es desarticula tota la operació i no cal seguir fent consultes.

En cas del correu:

```php
// Validar si el mail ja està registrat per qualsevol altre usuari
if ($email_ha_canviat && $error === '') {
    $stmt_check = $pdo->prepare(query: "SELECT id FROM usuaris WHERE email = ?");
    $stmt_check->execute(params: [$email]);
    if ($stmt_check->fetch()) {
        $error = "Aquest email ja està registrat per un altre usuari";
    }
}
```

En cas del telèfon:

```php
// Validar si el telèfon ja està registrat per qualsevol altre usuari
if ($telefon_ha_canviat && $error === '') {
    $stmt_tel = $pdo->prepare(query: "SELECT id FROM usuaris WHERE telefon = ?");
    $stmt_tel->execute(params: [$telefon]);
    if ($stmt_tel->fetch()) {
        $error = "Aquest telèfon ja està registrat";
    }
}
```

En cas del nom d'usuari

```php
// Validar si el nom d'usuari ja està registrat per qualsevol altre usuari
if ($usuari_ha_canviat && $error === '') {
    $stmt_user = $pdo->prepare(query: "SELECT id FROM usuaris WHERE nom_usuari = ?");
    $stmt_user->execute(params: [$nom_usuari]);
    if ($stmt_user->fetch()) {
        $error = "Aquest nom d'usuari ja està registrat";
    }
}
```

Per últim, en cas que la contrasenya també es modifiqui, verificarem:

- Que tots els camps estiguin emplenats
- Que la contrasenya actual sigui correcte amb la funció verificar_contrasenya, passant la contrasenya del formulari i el hash de la BD recuperat en la consulta inicial
- Que el camp de nova contrasenya i confirmació de nova contrasenya tinguin el mateix valor

```php
if ($canvi_contrasenya && $error === '') {
    if ($contrasenya_actual === '' || $nova_contrasenya === '' || $confirmar_contrasenya === '') {
        $error = "Has d'omplir tots els camps de contrasenya";
    } elseif (!verificar_contrasenya(contrasenya: $contrasenya_actual, hash: $usuari['contrasenya'])) {
        $error = "La contrasenya actual no és correcta";
    } elseif ($nova_contrasenya !== $confirmar_contrasenya) {
        $error = "Les noves contrasenyes no coincideixen";
    }
}
```

Cal remarcar que a l'HTML no especifiquem que els camps de contrasenya siguin requerits, ja que és possible que l'usuari no vulgui canviar la seva contrasenya, i en cas d'especificar els camps com a requerits, l'usuari hauria de canviar la contrasenya obligatòriament al enviar el formulari

Un cop s'han passat totes les verificacions de forma exitosa, és moment d'inserir les dades a la BD.

Com s'ha comentat anteriorment, farem un update de totes les dades, hagin estat o no canviades per evitar verificacions extenses, excepte per les contrasenyes, per tant:

- En cas que s'hagi modificat la contrasenya, farem un update de les dades personals i la contrasenya
- En cas que no s'hagi modificat la contrasenya, únicament actualitzarem les dades personals

Abans de res, hem de recordar que l'aplicació compta amb una verificació de correu electrònic, la qual haurà de canviar el seu estat a "no verificat" en cas de modificar el correu electrònic, en cas de no fer-ho, haurà de mantenir el seu valor anterior ("si" o "no").

Per això comprovarem si s'ha canviat l'email i assignarem el nou estat de verificació segons si hi ha hagut canvi o no

```php
// Si no hi ha errors
if ($error === '') {
    try {
        if ($email_ha_canviat) { // Si l'email ha canviat
            $nou_estat_verificacio = 'no';
        } else { // Si l'email no ha canviat
            $nou_estat_verificacio = $usuari['email_verificat']; // Valor original de l'usuari
        }
```

Un cop definit l'estat de verificació segons si s'ha modificat o no l'email, farem l'actualització de dades.

En cas que s'hagi modificat la contrasenya serà el següent:

```php
if ($canvi_contrasenya) {
    $hash = encriptar_contrasenya(contrasenya: $nova_contrasenya);
    $stmt = $pdo->prepare(query: "
        UPDATE usuaris
        SET nom_usuari = ?, nom_complet = ?, email = ?, telefon = ?, ciutat = ?, edat = ?, contrasenya = ?, email_verificat = ?
        WHERE id = ?
    ");
    $stmt->execute(params: [$nom_usuari, $nom_complet, $email, $telefon, $ciutat, $edat, $hash, $nou_estat_verificacio, $usuari_id]);
    registrar_activitat(pdo: $pdo, usuari_id: $usuari_id, accio: 'restablir-contrasenya');
```

En cas contrari, el següent:

```php
} else { // Si no canviem la contrasenya
    $stmt = $pdo->prepare(query: "
        UPDATE usuaris
        SET nom_usuari = ?, nom_complet = ?, email = ?, telefon = ?, ciutat = ?, edat = ?, email_verificat = ?
        WHERE id = ?
    ");
    // Únicament actualitzem les dades personals
    $stmt->execute(params: [$nom_usuari, $nom_complet, $email, $telefon, $ciutat, $edat, $nou_estat_verificacio, $usuari_id]);
}
```

Degut a que és possible que s'hagi modificat valors que la sessió fa servir, haurem d'actualitzar-los

```php
// Actualizar sessió
$_SESSION['usuari']['nom_usuari']   = $nom_usuari;
$_SESSION['usuari']['nom_complet']  = $nom_complet;
$_SESSION['usuari']['email']        = $email;
```

I modificarem les dades que hem obtingut de la consulta inicial per mostrar les dades actualitzades al formulari

```php
// Refrescar datos locales para el formulario
$usuari['nom_usuari']  = $nom_usuari;
$usuari['nom_complet'] = $nom_complet;
$usuari['email']       = $email;
$usuari['telefon']     = $telefon;
$usuari['ciutat']      = $ciutat;
$usuari['edat']        = $edat;
```

Per últim, en cas que l'email hagi canviat, esborrarem qualsevol codi de verificació que s'hagi desat a la sessió i mostrarem un missatge diferent al genèric, recordant que s'ha de tornar a verificar el correu electrònic

```php
if ($email_ha_canviat) {
    // Borrar código de verificación antiguo si existía para forzar uno nuevo
    unset($_SESSION['codi_verificacio']);
    $correcte = "Perfil actualitzat. Com que has canviat l'email, l'hauràs de tornar a verificar.";
} else {
    $correcte = "Perfil actualitzat correctament";
}
```

A més, registrarem una acció d'actualització del perfil

```php
registrar_activitat(pdo: $pdo, usuari_id: $usuari_id, accio: 'editar-perfil');
```

En cas que hagi succeït cap error o s'hagi realitzat la operació amb èxit, es mostrarà a la part superior del formulari

```php
<?php if ($error): ?>
    <div class="alert alert-error"><?= htmlspecialchars(string: $error) ?></div>
<?php endif; ?>

<?php if ($correcte): ?>
    <div class="alert alert-success"><?= htmlspecialchars(string: $correcte) ?></div>
<?php endif; ?>
```

Verificació de correu electrònic

L'aplicació compta amb un procés de verificació de correu electrònic. Aquest procés consta d'un enviament d'un codi de 8 dígits al correu especificat, i l'usuari ha d'introduir aquest codi dins del formulari de verificació.

És important remarcar que cal tenir el correu electrònic verificat per poder iniciar sessió amb 2FA.

Per això, iniciarem la sessió per recuperar les dades de l'usuari, com la seva ID, el seu rol o si està o no autenticat. A més recuperarem la connexió a la BD i el fitxer amb les funcions.

```php
<?php
session_start();
require 'funcions.php';
require './connexioBD/connexioRW.php';
```

Seguidament, recuperarem l'email i l'estat de verificació de l'email de l'usuari amb la ID obtinguda de la sessió. En cas que l'usuari no existeixi, retornarem a la pàgina de login. A més, en cas que l'estat de verificació de l'email sigui "si", retornarem a l'usuari a la seva pàgina privada, evitant que torni a passar pel procés de verificació

```php
// ID de l'usuari a partir de la sessió
$usuari_id = $_SESSION['usuari']['id'];

// Obtenir l'email i l'estat de l'email de l'usuari
$stmt = $pdo->prepare(query: "SELECT email, email_verificat FROM usuaris WHERE id = ?");
$stmt->execute(params: [$usuari_id]);
$usuari = $stmt->fetch();

if (!$usuari) {
    header(header: 'Location: login.php');
    exit;
}

if ($usuari["email_verificat"] == "si"){
    header(header: 'Location: privada.php');
    exit;
}
```

Seguidament, generarem un codi de verificació, però únicament si no existeix ja un. Així, si l'usuari refresca la pàgina, no tornarà a enviar el codi de nou, sinó que l'enviat anteriorment encara serà vàlid.

Tot i això, sempre que l'usuari surti d'aquesta pàgina sense haver especificat el codi de verificació, aquest serà esborrat.

Aquest codi es desarà a la sessió de l'usuari, per en cas que refresqui la pàgina i serà un nombre enter de 8 dígits

```php
// Generar y enviar código si aún no hay uno
if (!isset($_SESSION['codi_verificacio'])) {
    $_SESSION['codi_verificacio'] = random_int(min: 10000000, max: 99999999);
```

Seguidament, prepararem les capçaleres del correu:

- Destinatari: l'email de l'usuari
- Assumpte: Codi de verificació
- Missatge: "El codi generat anteriorment"
- Altres capçaleres: per complir amb stàndards de seguretat de Nominalia. Si un correu no té unes bones capçaleres, pot ser interpretat com spam

```php
// Generar y enviar un codi de verificació sempre que no hi hagi un a la sessió
if (!isset($_SESSION['codi_verificacio'])) {
    // Generem un codi de 8 xifres aleatori
    $_SESSION['codi_verificacio'] = random_int(min: 10000000, max: 99999999); // Desem el codi a la sessió

    // Enviament del correu
    $to = $usuari['email'];  // Destinatari: l'email de l'usuari registrat a la BD
    $subject = "Codi de verificació de correu electrònic"; // Assumpte del correu

    // Cos del missatge
    $message = "Hola {$usuari['nom_complet']},\n\nEl teu codi és: " . $_SESSION['codi_verificacio'] .  "\n\nSalutacions, l'equip de gserrat.cat";

    // Capçaleres genèriques per evitar ser categoritzat com a spam o correu sospitós
    $headers = [
        "From: no-reply@gserrat.cat",
        "Reply-To: no-reply@gserrat.cat",
        "MIME-Version: 1.0",
        "Content-Type: text/plain; charset=utf-8",
        "Date: " . date(format: "r"),
        "X-Mailer: PHP/" . phpversion()
    ];
```

A continuació, enviarem el correu amb la funció interna de PHP mail(). Aquesta funció espera rebre:

- Destinatari
- Assumpte
- Missatge
- Altres capçaleres

Per tant indicarem les variables anteriorment definides. En cas de les altres capçaleres, degut a que és una array amb diferents vectors, farem servir la funció interna de PHP implode(), la qual ajunta els elements d'una array com cadenes de text amb un separador indicat.

```php
    // Enviem l'email fent servir la funció mail() configurada a PHP.ini
    if (mail(to: $to, subject: $subject, message: $message, additional_headers: implode(s…"\r\n", $headers))) {
    } else {
        $errorMail = "Error al enviar el correu.";
    }
}
```

És molt important configurar PHP.ini per indicar a la funció mail quin servidor de correu electrònic local fer servir per l'enviament (Postfix, msmtp, etc). Per més informació sobre com configurar un servidor per l'enviament de correus electrònics es pot consultar l'apartat "Configuració del VPS per l'enviament de correus electrònics amb PHP"

Seguidament, el codi PHP esperarà que s'introdueixi el codi enviat al correu electrònic en un formulari. Al respondre el formulari amb el codi es comprovarà si el codi és correcte o no.

En cas que sigui correcte, actualitzarem la BD especificant que l'usuari ja té el correu electrònic verificat. A més, esborrarem el codi de verificació de la sessió i redirigirem a l'usuari a la seva pàgina privada.

En cas que no ho tingui, mostrarem un error i refrescarem la pàgina. Com hem desat el codi a la sessió i només es genera un de nou quan no existeix cap, tot i tenir un error i refrescar la pàgina, l'usuari podrà introduir el mateix codi generat al principi i no es generarà un codi nou cada cop que s'equivoqui.

```php
// Processar formulari
if ($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['codi_verificacio'])) {
    $codi = trim(string: $_POST['codi_verificacio']);

    if ($codi == $_SESSION['codi_verificacio']) {
        // Codi correcte, actualitzar BD
        $stmt = $pdo->prepare(query: "UPDATE usuaris SET email_verificat = 'si' WHERE id = ?");
        $stmt->execute(params: [$usuari_id]);

        unset($_SESSION['codi_verificacio']); // eliminar codi de la sessió
        header(header: 'Location: privada.php');
        exit;
    } else {
        $errorCodi = "El codi introduït és incorrecte.";
    }
}
```

Tauler de l'administrador

Els usuaris administradors tenen accés a un tauler d'administració, on surten diferents mètriques de l'aplicació.

Els usuaris que compten amb el rol d'administrador editor, a més podran restablir contrasenyes d'altres usuaris i desbloquejar comptes bloquejats.

Primerament, iniciarem la sessió per recuperar les dades de l'usuari, com la seva ID, el seu rol o si està o no autenticat. A més recuperarem la connexió a la BD i el fitxer amb les funcions

```php
<?php
session_start();
require 'funcions.php';
require './connexioBD/connexioRW.php';
```

En aquest cas no només comprovarem que l'usuari està autenticat amb la funció esta_autenticat, sinó que també comprovarem que sigui administrador, amb la funció es_admin

```php
// Verificar que està autenticat
requerir_autenticacio();

// Només els administradors són autoritzats
if (!es_admin()) {
    header(header: "Location: privada.php");
    exit;
}
```

A continuació farem una consulta SQL per recuperar les següents dades

- De tots els usuaris
  - Autenticacions exitoses
  - Autenticacions errònies
  - Restabliments de contrasenyes
- Dels usuaris normals
  - Autenticacions exitoses
  - Autenticacions errònies
  - Restabliments de contrasenyes
- Dels administradors (editors i visualitzadors)
  - Autenticacions exitoses
  - Autenticacions errònies
  - Restabliments de contrasenyes

```php
try{
    // Estadístiques globals
    $stmt = $pdo->query(query: "
        SELECT
            -- Autenticacions correctes (login + auto-login) de tothom
            SUM(a.accio IN ('login', 'auto-login')) AS total_autenticacions,

            -- Logins erronis de tothom
            SUM(a.accio = 'error-login') AS total_error_logins,

            -- Restabliments de contrasenyes
            SUM(a.accio = 'restablir-contrasenya') AS total_restabliment_contrasenyes,

            -- Autenticacions correctes d'usuaris normals
            SUM(
                a.accio IN ('login', 'auto-login')
                AND u.rol = 'usuari'
            ) AS auth_usuaris,

            -- Logins erronis d'usuaris normals
            SUM(
                a.accio = 'error-login'
                AND u.rol = 'usuari'
            ) AS errors_usuaris,

            -- Restabliment de contrasenyes d'usuaris normals
            SUM(
                a.accio = 'restablir-contrasenya'
                AND u.rol = 'usuari'
            ) AS restabliments_contrasenya_usuaris,
```

```php
            -- Autenticacions correctes d'administradors
            SUM(
                a.accio IN ('login', 'auto-login')
                AND u.rol IN ('ed_admin', 'vi_admin')
            ) AS auth_admins,

            -- Logins erronis d'administradors
            SUM(
                a.accio = 'error-login'
                AND u.rol IN ('ed_admin', 'vi_admin')
            ) AS errors_admins,

            -- Restabliment de contrasenyes d'administradors
            SUM(
                a.accio = 'restablir-contrasenya'
                AND u.rol IN ('ed_admin', 'vi_admin')
            ) AS restabliments_contrasenya_admins

        FROM activitat a
        JOIN usuaris u ON u.id = a.usuari_id
    ");
    $autenticacions = $stmt->fetch();
```

I les mostrarem al tauler. En cas que no hi hagi cap valor especificarem que surti un 0

```php
<section class="stats-section">
    <h2 class="section-title">Estadístiques globals</h2>
    <div class="stats-grid-top">
        <div class="stat-card main">
            <span class="label">Total Autenticacions</span>
            <span class="value"><?= $autenticacions['total_autenticacions'] ?? 0 ?></span>
        </div>
        <div class="stat-card error">
            <span class="label">Logins Erronis</span>
            <span class="value"><?= $autenticacions['total_error_logins'] ?? 0 ?></span>
        </div>
        <div class="stat-card warn">
            <span class="label">Restabliments</span>
            <span class="value"><?= $autenticacions['total_restabliment_contrasenyes'] ?? 0 ?></span>
        </div>
    </div>

    <div class="stats-grid-secondary">
        <div class="stat-group">
            <h3>Usuaris normals</h3>
            <div class="row"><span>Autenticacions:</span> <strong><?= $autenticacions['auth_usuaris'] ?? 0 ?></strong></div>
            <div class="row"><span>Errors d'autenticació:</span> <strong><?= $autenticacions['errors_usuaris'] ?? 0 ?></strong></div>
            <div class="row"><span>Restabliments de contrasenyes:</span> <strong><?= $autenticacions['restabliments_contrasenya_usuaris'] ?? 0 ?></strong></div>
        </div>
        <div class="stat-group">
            <h3>Administradors</h3>
            <div class="row"><span>Autenticacions:</span> <strong><?= $autenticacions['auth_admins'] ?? 0 ?></strong></div>
            <div class="row"><span>Errors d'autenticació:</span> <strong><?= $autenticacions['errors_admins'] ?? 0 ?></strong></div>
            <div class="row"><span>Restabliments de contrasenyes:</span> <strong><?= $autenticacions['restabliments_contrasenya_admins'] ?? 0 ?></strong></div>
        </div>
```

En cas que l'administrador sigui editor, podrà veure un tauler més extens. Per això realitzem un condicional que comprovi que sigui un administrador editor amb la funció es_admin_ed

```php
<?php if (es_admin_ed()): ?>
```

Per començar, l'administrador editor pot esborrar totes les estadístiques anteriorment comentades mitjançant un botó.

Quan es prem, es realitza una consulta SQL per esborrar tots els registres de la taula activitat.

```php
// Borrar estadístiques
if ($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['borrar_activitat']) && es_admin_ed()) {
    try {
        $stmt = $pdo->prepare(query: "DELETE FROM activitat");
        $stmt->execute();
        header(header: "Location: taulerAdmin.php");
        exit;
    } catch (PDOException $e) {
        $error_reset = "No s'ha pogut esborrar l'historial. Motiu: " . $e->getMessage();
    }
}
```

Seguidament s'incorpora un formulari simple de restabliment de contrasenya, amb una entrada per especificar l'usuari i una altre per la contrasenya.

Al prémer el botó de restablir contrasenya, verifiquem que els dos camps tenen algun valor

```php
if ($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['usuari_nom']) && es_admin_ed()) {
    $usuari_nom = trim(string: $_POST['usuari_nom']);

    if ($usuari_nom === '') {
        $error_reset = "Indica un nom d'usuari.";
    } elseif (empty($_POST['nova_contrasenya'])) {
        $error_reset = "Indica una contrasenya.";
```

A continuació, realitzem una consulta SQL per comprovar si l'usuari existeix

```php
} else {
    try {
        // Buscar usuari per nom d'usuari (únic)
        $stmt = $pdo->prepare(query: "SELECT id FROM usuaris WHERE nom_usuari = ?");
        $stmt->execute(params: [$usuari_nom]);
        $usuari = $stmt->fetch();

        if (!$usuari) {
            $error_reset = "Usuari no trobat.";
```

Si l'usuari existeix, recuperarem i encriptarem la contrasenya, actualitzarem l'usuari amb la nova contrasenya i modificarem els intents d'inici de sessió a l'usuari a 0, per així desbloquejar el compte també

```php
} else {
    // Si l'usuari existeix, actualitzem contrasenya i nombre d'intents
    if (isset($_POST['nova_contrasenya']) && $_POST['nova_contrasenya'] !== '') {
        // Recuperem la contrasenya del formulari
        $nova_contrasenya = trim(string: $_POST['nova_contrasenya']);

        // Encriptem la constrasenya
        $hash = password_hash(password: $nova_contrasenya, algo: PASSWORD_DEFAULT);

        // Inserim la nova contrasenya
        $stmt = $pdo->prepare(query: "UPDATE usuaris SET contrasenya = ? WHERE id = ?");
        $stmt->execute(params: [$hash, $usuari['id']]);

        // Reiniciem el número d'intents a 0, així també desbloquejem l'usuari
        $stmt = $pdo->prepare(query: "UPDATE usuaris SET intents_login = 0, ultim_intent = NULL WHERE id = ?");
        $stmt->execute(params: [$usuari['id']]);
```

Per últim, definim el missatge d'èxit i registrem l'acció a la taula d'activitat com a "restablir-contrasenya" amb la funció registrar_activitat

Com a última funcionalitat de l'administrador editor, es permet desbloquejar comptes d'usuaris sense la necessitat de restablir la contrasenya.

Aquesta funcionalitat únicament requereix especificar el nom d'usuari a desbloquejar.

Quan es prem el botó de desbloquejar usuari, a l'igual que per restablir la contrasenya, es verifica que el camp tingui un valor i que l'usuari realment existeixi

```php
// Desbloqueig d'usuari

if ($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['usuari_nom_desbloquejar']) && es_admin_ed()) {
    $usuari_nom = trim(string: $_POST['usuari_nom_desbloquejar']);

    if ($usuari_nom === '') {
        $error_reset = "Omple el nom d'usuari per desbloquejar.";
    } else {
        try {
            // Buscar usuari per nom
            $stmt = $pdo->prepare(query: "SELECT id FROM usuaris WHERE nom_usuari = ?");
            $stmt->execute(params: [$usuari_nom]);
            $usuari = $stmt->fetch();

            if (!$usuari) {
                $error_reset = "Usuari no trobat.";
```

Si l'usuari existeix, especificarem els seus intents d'inici de sessió a 0 per així desbloquejar-lo. A més especificarem el missatge d'èxit i registrem l'acció a la taula d'activitat com a "desbloquejar-compte" amb la funció registrar_activitat.

```php
} else {
    // Si existeix, definim el nombre d'intents de login a 0 i registrem l'acció "desbloquejar-compte"
    $stmt = $pdo->prepare(query: "UPDATE usuaris SET intents_login = 0, ultim_intent = NULL WHERE id = ?");
    $stmt->execute(params: [$usuari['id']]);
    $missatgeDesbloqueig = "Compte desbloquejat correctament.";
    registrar_activitat(pdo: $pdo, usuari_id: $usuari['id'], accio: 'desbloquejar-compte');
}
```

Logout

En el moment que l'usuari tanca la sessió, es registra a la BD una activitat conforme s'ha fet un logout, es destrueix la sessió existent i s'esborren les cookies.

Primerament, iniciarem la sessió per recuperar les dades de l'usuari, com la seva ID, el seu rol o si està o no autenticat. També requerirem la connexió a la BD i el fitxer amb les funcions

```php
session_start();
require 'funcions.php';
require './connexioBD/connexioRW.php';
```

Seguidament, comprovarem que realment hi ha un usuari que ha iniciat sessió. En cas que s'accedeixi a la pàgina sense haver iniciat sessió, es redirigirà a la pàgina d'inici de sessió

```php
if (isset($_SESSION['usuari'])) {
    registrar_activitat(pdo: $pdo, usuari_id: $_SESSION['usuari']['id'], accio: 'logout');
} else {
    header(header: 'Location: login.php');
    exit;
}
```

En cas que existeixi l'usuari, esborrarem la sessió i les cookies i retornarem l'usuari a l'inici

```php
// Netejar sessions i cookies
$_SESSION = [];
session_destroy();
setcookie(name: 'recordar_id', value: '', expires_or_options: time() - 3600, path: '/');
setcookie(name: 'recordar_token', value: '', expires_or_options: time() - 3600, path: '/');
if (isset($_COOKIE[session_name()])) {
setcookie(name: session_name(), value: '', expires_or_options: time()-3600, path: '/');
}

// Redirigim l'usuari a index.php
header(header: 'Location: index.php');
exit;
```

Funcions

encriptar_contrasenya

Aquesta funció rep una contrasenya i retorna aquesta mateixa contrasenya encriptada (hash)

Durant l'enregistrament d'usuaris, modificació de perfil o restabliments de contrasenya, l'usuari introdueix una contrasenya en text pla que s'ha de desar a la BD, i per questions òbvies de seguretat, no es pot desar una contrasenya en text pla, sinó que s'ha d'encriptar

Per encriptar la contrasenya es fa servir la funció interna de PHP password_hash on els seus paràmetres són la contrasenya a encriptar i el mètode d'encriptació, en aquest cas, el mètode per defecte

```php
// Encriptar contrasenya
2 references
function encriptar_contrasenya($contrasenya): string {
    return password_hash(password: $contrasenya, algo: PASSWORD_DEFAULT);
}
```

verificar_contrasenya

Aquesta funció rep una contrasenya i un hash i retorna aquesta un valor boleà depenent si la contrasenya coincideix amb el hash o no.

Durant l'autenticació d'usuaris, l'usuari introdueix la seva contrasenya en text pla, però la contrasenya desada a la BD no és en text pla, sinó que és un hash

Per comprovar si la contrasenya correspon a un hash es fa servir la funció interna de PHP password_verify on els seus paràmetres són la contrasenya en text pla i el hash.

```php
// Verificar contrasenya
2 references
function verificar_contrasenya($contrasenya, $hash): bool {
    return password_verify(password: $contrasenya, hash: $hash);
}
```

registrar_activitat

Aquesta funció rep la connexió de la BD, la ID de l'usuari i l'acció realitzada. Aquesta funció no retorna cap valor ja que el seu objectiu és actualitzar la BD.

Durant l'execució de l'aplicació, es realitzaran diferents accions que es desaran a una taula de la BD registrant la data, la IP de l'equip i el tipus d'acció realitzada.

Per això, executarem un Insert a la BD amb les dades que hem rebut.

```php
// Registrar activitat
12 references
function registrar_activitat($pdo, $usuari_id, $accio): void {
    $ip = $_SERVER['REMOTE_ADDR'];
    $stmt = $pdo->prepare("INSERT INTO activitat (usuari_id, accio, ip_client) VALUES (?, ?, ?)");
    $stmt->execute([$usuari_id, $accio, $ip]);
```

esta_autenticat

Aquesta funció no rep cap paràmetre i retorna un valor boleà depenent de si l'usuari està autenticat o no.

Per evitar que usuaris no autenticats puguin accedir a pàgines privades, hem de verificar que l'usuari estigui autenticat.

Per això, comprovarem que s'ha iniciat una sessió i que el vector "autenticat" de l'usuari de la sessió sigui verdader

```php
// Comprobar si l'usuari està autenticat
4 references
function esta_autenticat(): bool {
    return isset($_SESSION['usuari']) && $_SESSION['usuari']['autenticat'] === true;
}
```

requerir_autenticacio

Aquesta funció no rep cap paràmetre ni retorna cap valor.

Mentre que la funció esta_autenticat únicament retorna si l'usuari està o no autenticat, aquesta funció comprova si ho està o no, i en cas negatiu, redirigeix l'usuari a la pàgina de login.

Per això comprovem que resultat de la funció esta_autenticat sigui fals (és a dir, no està autenticat), i enviem l'usuari a la pàgina de login

```php
// Protegir pàgines privades
4 references
function requerir_autenticacio(): void {
    if (esta_autenticat() == false) {
        header(header: 'Location: login.php');
        exit;
    }
}
```

es_admin

Aquesta funció no rep cap paràmetre i retorna un valor boleà depenent si l'usuari és administrador o no.

A l'aplicació existeix un espai privat que només els administradors poden accedir, és per això que abans de permetre l'entrada a un usuari, hem d'assegurar-nos que sigui administrador, tant visualitzador com editor.

Per això, recuperem el vector "rol" de la sessió, que indica el tipus d'usuari que és, i comprovem si el valor és "ed_admin" o "vi_admin"

```php
// Comprovar si l'usuari és administrador
2 references
function es_admin(): bool {
    return isset($_SESSION['usuari']['rol']) && in_array(needle: $_SESSION['usuari']['rol'], haystack: ['ed_admin', 'vi_admin'], strict: true);
}
```

es_admin_ed

Aquesta funció no rep cap paràmetre i retorna un valor boleà depenent si l'usuari és administrador editor o no.

Dins de l'espai privat dels administradors, existeix un apartat on es pot modificar les contrasenyes d'altres usuaris i desbloquejar comptes. Aquesta secció únicament pot tenir accés l'administrador editor.

Per això, recuperem el vector "rol" de la sessió, que indica el tipus d'usuari que és, i comprovem si el valor és "ed_admin" exclusivament.

```php
// Comprovar si l'usuari és administrador editor
4 references
function es_admin_ed(): bool {
    return isset($_SESSION['usuari']['rol']) && ($_SESSION['usuari']['rol'] === 'ed_admin');
}
```

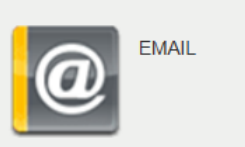
Configuració del VPS per l'enviament de correus electrònics amb PHP

Configuració de Nominalia

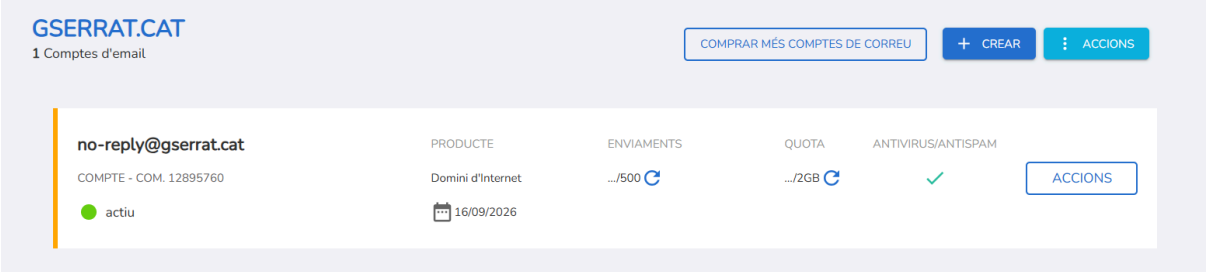
Degut a que Azure, a l'igual que molts ISP, bloqueja el port 25 per l'SMTP, haurem de fer servir un relay de correu electrònic per poder enviar emails a entitats externes.

Degut a que el meu domini, gserrat.cat, està proveït per Nominalia, faré servir el seu servidor de correu electrònic com a relay.

Per això, hauré de crear un compte de correu dins del servei de Nominalia. Per això, dins del tauler de control, cercarem l'opció de configuració d'email



Dins d'aquesta configuració, ens permetrà crear comptes de correu electrònic. En aquest cas he creat el compte no-reply@gserrat.cat



És molt important recuperar les dades del servidor de correu sortint SMTP de Nominalia per poder indicar al VPS que volem que els correus s'enviïn amb el compte de no-reply.

Podem dirigir-nos a la pàgina oficial de Nominalia sobre els seus servidors i recuperar aquestes dades.

En cas de tenir un proveïdor de dominis diferent, tindrà unes dades alternatives que s'haurà de consultar en cada cas

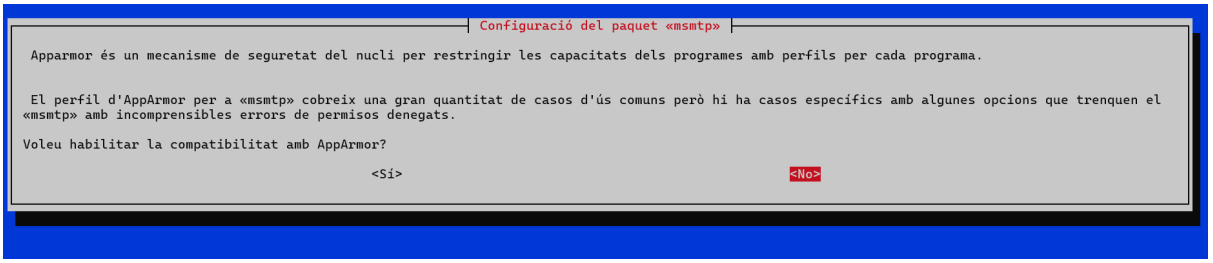
Dades del Servidor de Correu Sortint (SMTP)

- **Servidor o Host:** authsmtp.securemail.pro
- **Usuari:** El teu compte de correu. (Per exemple: *tucuenta@esteesmidominio.com*)
- **Contrasenya:** La contrasenya que vas assignar al compte de correu.
- **Port SMTP:** 465
- **Tipus de Seguretat:** SSL

Configuració del VPS

Dins del VPS, instal·larem el client SMTP anomenat msmtp, el qual està pensat per redirigir els correus a un servidor SMTP específic, que s'encarregarà de l'enviament.

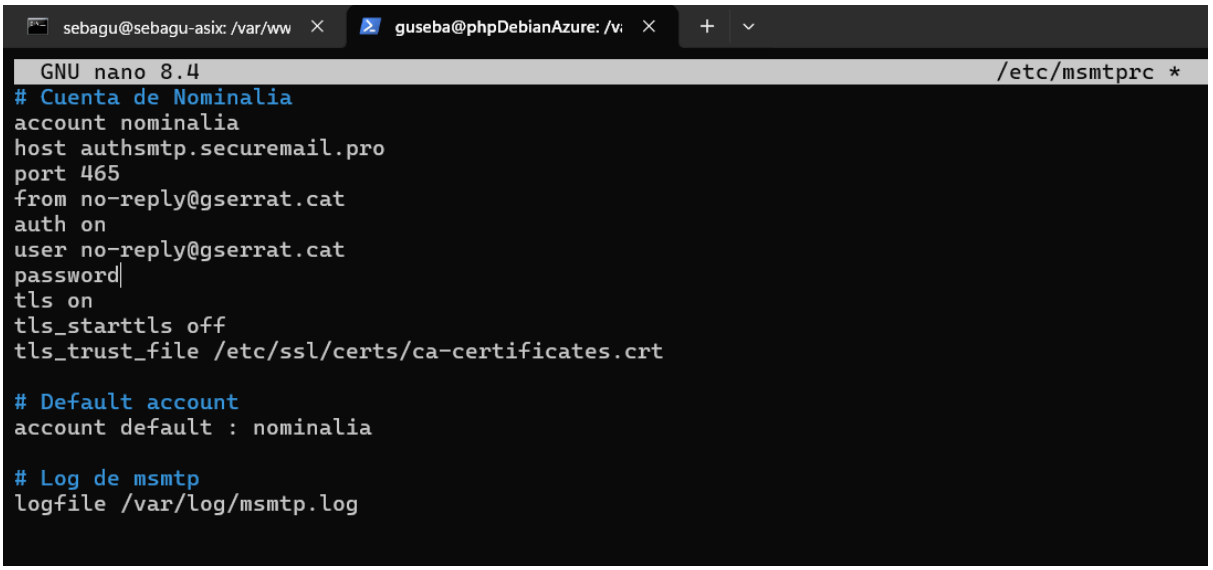
Per instal·lar msmtp únicament caldrà executar l'ordre *sudo apt install msmtp.* Quan ens pregunti si volem habilitar AppArmor indicarem que no



Un cop instal·lat, configurarem el client per indicar a quin servidor SMTP haurem d'enviar els correus sortints. El fitxer de configuració pot estar en qualsevol directori, ja que més endavant especificarem a PHP on és, però generalment els fitxers de configuració es troben a /etc.

Dins del fitxer, inclourem la configuració del servidor de correu SMTP del nostre proveïdor de domini, en cas de Nominalia, es pot consultar en la seva pàgina oficial

En aquest cas, crearé el fitxer /etc/msmtp amb el següent contingut:



Degut a que el que realitza l'enviament de correu és la pàgina web, l'usuari intern del sistema que realment realitza l'acció és www-data. Per això modificarem la propietat de l'arxiu de configuració a aquest usuari amb l'ordre
sudo chown www-data:www-data /etc/msmtprc

Seguidament, msmtp requereix que únicament el propietari de l'arxiu tingui permisos, ja que s'està desant les credencials en text pla. La modificació de permisos es realitzarà amb la següent ordre *sudo chmod 600 /etc/msmtprc*

Per últim, comprovem que els permisos estan definits correctament

```
guseba@phpDebianAzure:/etc$ sudo nano /etc/msmtprc
guseba@phpDebianAzure:/etc$ ls -lsa /etc/msmtprc
4 -rw------- 1 www-data www-data 324 Jan  4 17:51 /etc/msmtprc
guseba@phpDebianAzure:/etc$
```

Al fitxer de configuració del client smtp, hem definit un fitxer de log on desar les sortides de l'aplicació. A l'igual que amb el fitxer de configuració, hem de definir la propietat a l'usuari www-data per que pugui escriure, a més de crear el propi fitxer.

```
guseba@phpDebianAzure:/var/log$ ls -ls msmtp.log
0 -rw-rw---- 1 www-data www-data 0 Jan  4 17:45 msmtp.log
guseba@phpDebianAzure:/var/log$
```

Configuració de PHP

Per defecte, quan PHP executa la funció mail(), envia un correu amb el servidor sendmail. Per canviar això, hem de modificar la configuració de PHP i especificar un nou servidor de correu sortint.

Per canviar el servidor sortint de PHP per defecte hem de modificar el fitxer php.ini a la següent ruta: */etc/php/8.4/apache2/php.ini,* i canviar el valor de la directiva "sendmail_path" a *"/usr/bin/msmtp -C /etc/msmtprc -t"*

- */usr/bin/msmtp* indica a PHP on és el binari de l'aplicació a fer servir
- *-C /etc/msmtprc* indica a PHP quin és el fitxer de configuració que ha de fer servir
- *-t* és per indicar al servidor msmtp que llegeixi els destinataris a partir de les capçaleres del missatge rebudes de PHP.

```
; For Unix only.  You may supply arguments as well (default: "sendmail -t -i").
; https://php.net/sendmail-path
sendmail_path = "/usr/bin/msmtp -C /etc/msmtprc -t"
```

Un cop configurat el fitxer .ini de PHP, caldrà reiniciar el servidor web

```
guseba@phpDebianAzure:/etc/php/8.4/apache2$ sudo systemctl restart apache2
guseba@phpDebianAzure:/etc/php/8.4/apache2$
```

Per últim, es pot consultar en fitxer de configuració de PHP per verificar que el valor de la directiva és correcte

Directive	Local Value	Master Value
sendmail_path	/usr/bin/msmtp -C /etc/msmtprc -t	/usr/bin/msmtp -C /etc/msmtprc -t

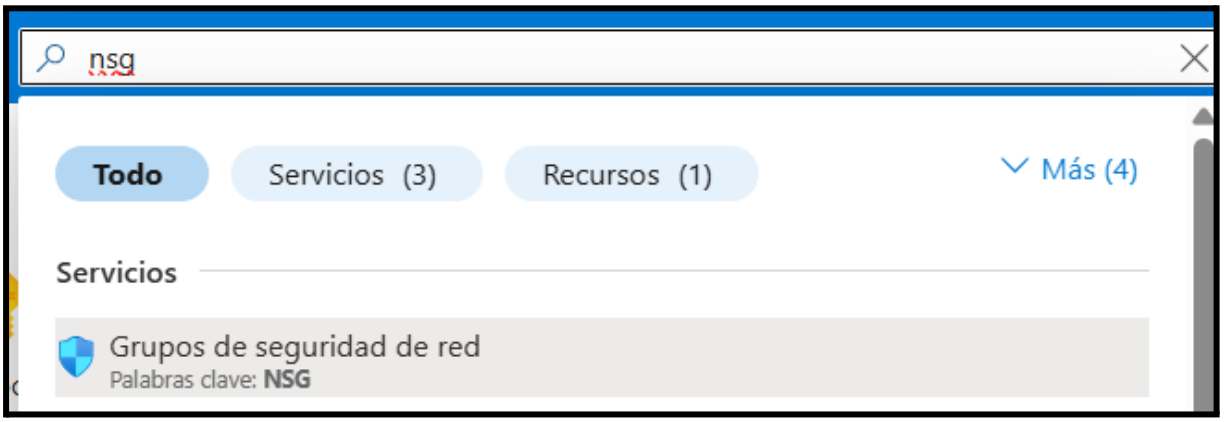
Configuració del NSG d'Azure

Azure, la plataforma de Cloud que allotja el meu VPS, compta amb l'implementació de firewalls anomenats Network Security Groups (NSG) en les interfícies de xarxa de VMs o VNets.

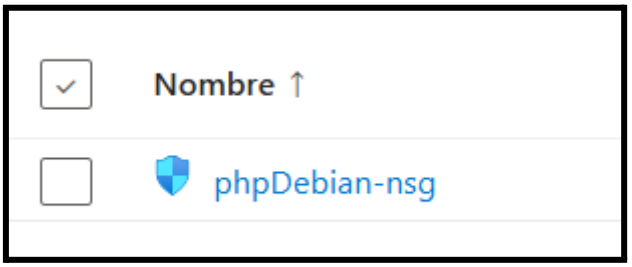
En cas de voler permetre la comunicació entrant o sortint d'algun port en específic, cal definir una ACL, ja sigui d'entrada o de sortida.

En aquest cas, per la comunicació entre el client msmtp del VPS i el servidor SMTP de Nominalia estem requerint una comunicació sortint al port 465, i per tant hem d'habilitar una ACL que permeti la sortida de comunicacions pel port 465.

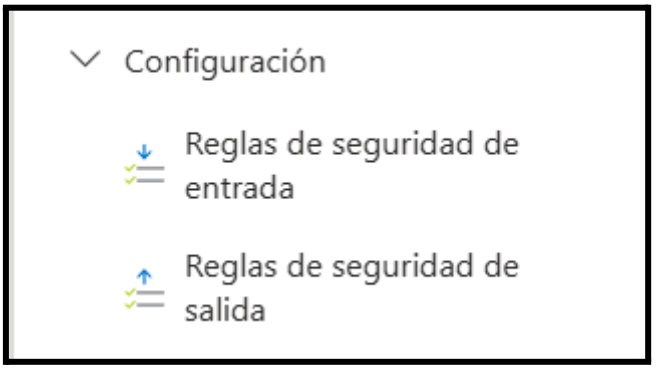
Per això, dins d'Azure, cercarem el servei d'NSGs



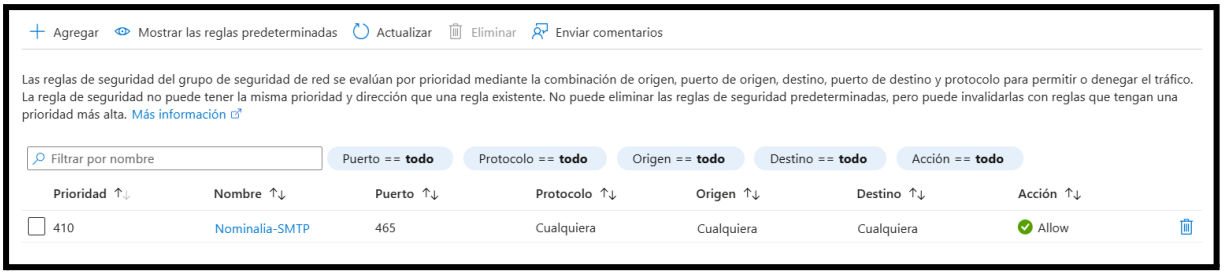
Dins del servei d'NSGs, trobarem un llistat de tots els NSGs creats en la nostra subscripció d'Azure. En el meu cas únicament tinc un de la màquina virtual de PHP



Al seleccionar l'NSG, a la barra lateral esquerra trobarem un apartat anomenat "Configuració", on podrem definir regles d'entrada i de sortida



Un cop dins de les regles de sortida, crearem una regla pel port 465 per qualsevol protocol, qualsevol origen i qualsevol destí



I un cop creada aquesta regla, tot el tràfic sortint pel port 465 serà permès per Azure.

També es pot consultar la següent video de com configurar un NSG a Azure
nsg.mp4